

个人资料



splenday1989

访问: 41503次

积分: 651

等级:

排名: 千里之外

原创: 10篇 转载: 89篇

译文: 2篇 评论: 1条

文章搜索

文章分类

loadRunner (9)

MongoDb (5)

Linux (75)

Python (3)

Docker (1)

QTP (5)

文章存档

2016年04月 (1)

2016年03月 (4)

2016年02月 (3)

2016年01月 (2)

2015年12月 (4)

展开

阅读排行

Samba配置详解 (7021)

Shell脚本中参数传递方法 (6039)

iptables基础知识详解 (3343)

LoadRunner Controller (1556)

Linux、Windows时间同步 (1186)

LoadRunner11+Win7+I (1021)

redhat 查询端口占用 (957)

supervisor监控mongoD (814)

基于linux的bond技术 (693)

每天一个Linux命令-10(L (683)

iptables基础知识详解

2016-02-19 14:42

3348人阅读

评论(1)

收藏

举报

分类: Linux (74)

iptables防火墙可以用于创建过滤(filter)与NAT规则。所有Linux发行版都能使用iptables, 因此理解如何配置iptables将会帮助你更有效地管理Linux防火墙。如果你是第一次接触iptables, 你会觉得它很复杂, 但是一旦你理解iptables的工作原理, 你会发现其实它很简单。

首先介绍iptables的结构: iptables -> Tables -> Chains -> Rules. 简单地讲, tables由chains组成, 而chains又由

TABLE 1

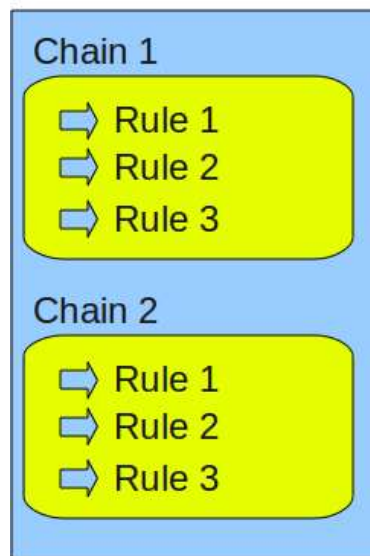
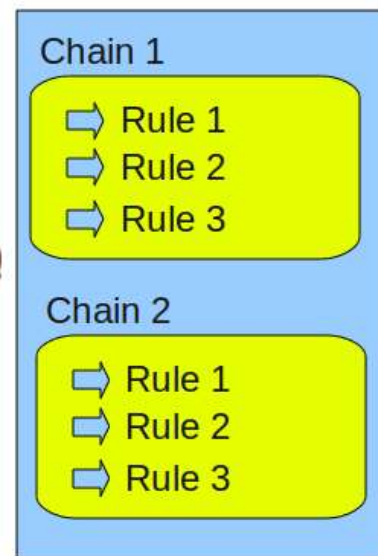


TABLE 2



rules组成。如下图所示。

图: IPTables Table, Chain, and Rule Structure

一、iptables的表与链

iptables具有Filter, NAT, Mangle, Raw四种内建表:

1. Filter表

Filter表示iptables的默认表, 因此如果你没有自定义表, 那么就默认使用filter表, 它具有以下三种内建链:

INPUT链 - 处理来自外部的数据。

OUTPUT链 - 处理向外发送的数据。

FORWARD链 - 将数据转发到本机的其他网卡设备上。

2. NAT表

NAT表有三种内建链:

PREROUTING链 - 处理刚到达本机并在路由转发前的数据包。它会转换数据包中的目标IP地址 (destination ip address), 通常用于DNAT(destination NAT)。

POSTROUTING链 - 处理即将离开本机的数据包。它会转换数据包中的源IP地址 (source ip address), 通常用于SNAT (source NAT)。

OUTPUT链 - 处理本机产生的数据包。

3. Mangle表

Mangle表用于指定如何处理数据包。它能改变TCP头中的QoS位。Mangle表具有5个内建链:

PREROUTING

OUTPUT

FORWARD

INPUT

POSTROUTING

评论排行

iptables基础知识详解 (1)
每天一个Linux命令-3 (l (0)
mongoVue的使用 (0)
SHELL正则表达式介绍 (0)
每天一个Linux命令-2 (v (0)
每天一个Linux命令-1 (g (0)
Linux系统启动过程详解 (0)
linux防火墙 (0)
ulimit 设置 (0)
linux命令（文件切割） (0)

推荐文章

* CSDN日报20170703——《从高考到程序员——我一直在寻找答案》
* 从源码剖析PopupWindow 兼容Android 6.0以上版本点击外部不消失
* 轻松学，Java 中的代理模式及动态代理
* ArcGIS水文分析实战教程——河流提取与河网分级
* Tensorflow中使用TFRecords高效读取数据--结合NLP数据实践
* 每周荐书：MyBatis、并行编程、Ansible（评论送书）

最新评论

iptables基础知识详解
zhushuangyin: 我尽量以通俗易懂的方式总结了iptables的概念，欢迎交流http://www.zsythink....

4. Raw表

Raw表用于处理异常，它具有2个内建链：

PREROUTING chain

OUTPUT chain

5.小结

下图展示了iptables的三个内建表：

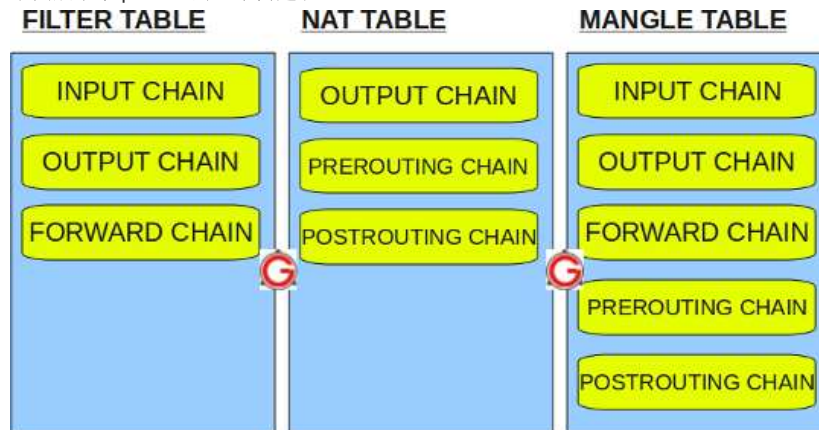


图: IPTables 内建表

二、IPTABLES 规则(Rules)

牢记以下三点式理解iptables规则的关键：

Rules包括一个条件和一个目标(target)

如果满足条件，就执行目标(target)中的规则或者特定值。

如果不满足条件，就判断下一条Rules。

目标值（Target Values）

下面是你可以在target里指定的特殊值：

ACCEPT – 允许防火墙接收数据包

DROP – 防火墙丢弃包

QUEUE – 防火墙将数据包移交到用户空间

RETURN – 防火墙停止执行当前链中的后续Rules，并返回到调用链(the calling chain)中。

如果你执行iptables -list你将看到防火墙上的可用规则。下例说明当前系统没有定义防火墙，你可以看到，它显示了默认的filter表，以及表内默认的input链, forward链, output链。

```
# iptables -t filter -list
```

```
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
```

查看mangle表：

```
# iptables -t mangle -list
```

查看NAT表：

```
# iptables -t nat -list
```

查看RAW表：

```
# iptables -t raw -list
```

!注意：如果不指定 -t选项，就只会显示默认的 filter表。因此，以下两种命令形式是一个意思：

```
# iptables -t filter -list
```

(or)

```
# iptables -list
```

以下例子表明在filter表的input链, forward链, output链中存在规则：

```
# iptables -list
```

```
Chain INPUT (policy ACCEPT)
```

```
num target prot opt source destination
1 RH-Firewall-1-INPUT all — 0.0.0.0/0 0.0.0.0/0
```

```
Chain FORWARD (policy ACCEPT)
```

```
num      target      prot  opt    source      destination
1    RH-Firewall-1-INPUT  all  -      0.0.0.0/0    0.0.0.0/0

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

Chain RH-Firewall-1-INPUT (2 references)
num target  prot opt source      destination
1  ACCEPT  all  -  0.0.0.0/0    0.0.0.0/0
2  ACCEPT  icmp -  0.0.0.0/0    0.0.0.0/0    icmp type 255
3  ACCEPT  esp  -  0.0.0.0/0    0.0.0.0/0
4  ACCEPT  ah  -  0.0.0.0/0    0.0.0.0/0
5  ACCEPT  udp  -  0.0.0.0/0    224.0.0.251    udp dpt:5353
6  ACCEPT  udp  -  0.0.0.0/0    0.0.0.0/0    udp dpt:631
7  ACCEPT  tcp  -  0.0.0.0/0    0.0.0.0/0    tcp dpt:631
8  ACCEPT  all  -  0.0.0.0/0    0.0.0.0/0    state RELATED,ESTABLISHED
9  ACCEPT  tcp  -  0.0.0.0/0    0.0.0.0/0    state NEW tcp dpt:22
10 REJECT  all  -  0.0.0.0/0    0.0.0.0/0    reject-with icmp-host-prohibited
```

以上输出包含下列字段：

num – 指定链中的规则编号
target – 前面提到的**target**的特殊值
prot – 协议：**tcp**, **udp**, **icmp**等
source – 数据包的源IP地址
destination – 数据包的目标IP地址

三、清空所有iptables规则

在配置iptables之前，你通常需要用**iptables -list**命令或者**iptables-save**命令查看有无现存规则，因为有时需要删除现有的iptables规则：

```
iptables -flush
```

或者

```
iptables -F
```

这两条命令是等效的。但是并非执行后就万事大吉了。你仍然需要检查规则是不是真的清空了，因为有的linux发行版上这个命令不会清除NAT表中的规则，此时只能手动清除：

```
iptables -t NAT -F
```

四、永久生效

当你删除、添加规则后，这些更改并不能永久生效，这些规则很有可能在系统重启后恢复原样。为了让配置永久生效，根据平台的不同，具体操作也不同。下面进行简单介绍：

1.Ubuntu

首先，保存现有的规则：

```
iptables-save > /etc/iptables.rules
```

然后新建一个bash脚本，并保存到 **/etc/network/if-pre-up.d/** 目录下：

```
#!/bin/bash
```

```
iptables-restore < /etc/iptables.rules
```

这样，每次系统重启后iptables规则都会被自动加载。

！注意：不要尝试在**.bashrc**或者**.profile**中执行以上命令，因为用户通常不是**root**，而且这只能在登录时加载iptables规则。

2.CentOS, RedHat

```
# 保存iptables规则
```

```
service iptables save
```

```
# 重启iptables服务
```

```
service iptables stop
```

```
service iptables start
```

查看当前规则：

```
cat /etc/sysconfig/iptables
```

五、追加iptables规则

可以使用**iptables -A**命令追加新规则，其中**-A**表示**Append**。因此，新的规则将追加到链尾。

一般而言，最后一条规则用于丢弃(DROP)所有数据包。如果你已经有这样的规则了，并且使用**-A**参数添加新规则

，那么就是无用功。

1.语法

iptables -A chain firewall-rule

-A chain — 指定要追加规则的链

firewall-rule — 具体的规则参数

2.描述规则的基本参数

以下这些规则参数用于描述数据包的协议、源地址、目的地址、允许经过的网络接口，以及如何处理这些数据包。

这些描述是对规则的基本描述。

-p 协议（protocol）

指定规则的协议，如tcp, udp, icmp等，可以使用all来指定所有协议。

如果不指定-p参数，则默认是all值。这并不明智，请总是明确指定协议名称。

可以使用协议名(如tcp)，或者是协议值（比如6代表tcp）来指定协议。映射关系请查看/etc/protocols

还可以使用--protocol参数代替-p参数

-s 源地址（source）

指定数据包的源地址

参数可以使IP地址、网络地址、主机名

例如：-s 192.168.1.101指定IP地址

例如：-s 192.168.1.10/24指定网络地址

如果不指定-s参数，就代表所有地址

还可以使用--src或者--source

-d 目的地址（destination）

指定目的地址

参数和-s相同

还可以使用--dst或者--destination

-j 执行目标（jump to target）

-j代表"jump to target"

-j指定了当与规则(Rule)匹配时如何处理数据包

可能的值是ACCEPT, DROP, QUEUE, RETURN, MASQUERADE

还可以指定其他链（Chain）作为目标

注：MASQUERADE，地址伪装，算是snat中的一种特例，可以实现自动化的snat（详情见上一篇文章）。

-i 输入接口（input interface）

-i代表输入接口(input interface)

-i指定了要处理来自哪个接口的数据包

这些数据包即将进入INPUT, FORWARD, PREROUTE链

例如：-i eth0指定了要处理经由eth0进入的数据包

如果不指定-i参数，那么将处理进入所有接口的数据包

如果出现! -i eth0，那么将处理所有经由eth0以外的接口进入的数据包

如果出现-i eth+，那么将处理所有经由eth开头的接口进入的数据包

还可以使用--in-interface参数

-o 输出（out interface）

-o代表"output interface"

-o指定了数据包由哪个接口输出

这些数据包即将进入FORWARD, OUTPUT, POSTROUTING链

如果不指定-o选项，那么系统上的所有接口都可以作为输出接口

如果出现! -o eth0，那么将从eth0以外的接口输出

如果出现-i eth+，那么将仅从eth开头的接口输出

还可以使用--out-interface参数

3.描述规则的扩展参数

对规则有了一个基本描述之后，有时候我们还希望指定端口、TCP标志、ICMP类型等内容。

--sport 源端口（source port）针对 -p tcp 或者 -p udp

缺省情况下，将匹配所有端口

可以指定端口号或者端口名称，例如"--sport 22"与"--sport ssh"。

/etc/services文件描述了上述映射关系。

从性能上讲，使用端口号更好

使用冒号可以匹配端口范围，如"--sport 22:100"

还可以使用"--source-port"

--dport 目的端口（**destination port**）针对 **-p tcp** 或者 **-p udp**

参数和**--sport**类似

还可以使用"**--destination-port**"

--tcp-flags **TCP**标志 针对 **-p tcp**

可以指定由逗号分隔的多个参数

有效值可以是：SYN, ACK, FIN, RST, URG, PSH

可以使用**ALL**或者**NONE**

--icmp-type **ICMP**类型 针对 **-p icmp**

--icmp-type 0 表示Echo Reply

--icmp-type 8 表示Echo

4.追加规则的完整实例：仅允许**SSH**服务

本例实现的规则将仅允许**SSH**数据包通过本地计算机，其他一切连接（包括**ping**）都将被拒绝。

1.清空所有**iptables**规则

```
iptables -F
```

2.接收目标端口为**22**的数据包

```
iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
```

3.拒绝所有其他数据包

```
iptables -A INPUT -j DROP
```

六、更改默认策略

上例的例子仅对接收的数据包过滤，而对于要发送出去的数据包却没有任何限制。本节主要介绍如何更改链策略，以改变链的行为。

1. 默认链策略

!\警告：请勿在远程连接的服务器、虚拟机上测试！

当我们使用**-L**选项验证当前规则是发现，所有的链旁边都有 **policy ACCEPT**标注，这表明当前链的默认策略为**ACCEPT**：

```
# iptables -L
```

```
Chain INPUT (policy ACCEPT)
```

```
target prot opt source destination
```

```
ACCEPT tcp -- anywhere anywhere tcp dpt:ssh
```

```
DROP all -- anywhere anywhere
```

```
Chain FORWARD (policy ACCEPT)
```

```
target prot opt source destination
```

```
Chain OUTPUT (policy ACCEPT)
```

```
target prot opt source destination
```

这种情况下，如果没有明确添加**DROP**规则，那么默认情况下将采用**ACCEPT**策略进行过滤。除非：

a)为以上三个链单独添加**DROP**规则：

```
iptables -A INPUT -j DROP
```

```
iptables -A OUTPUT -j DROP
```

```
iptables -A FORWARD -j DROP
```

b)更改默认策略：

```
iptables -P INPUT DROP
```

```
iptables -P OUTPUT DROP
```

```
iptables -P FORWARD DROP
```

糟糕！！如果你严格按照上一节的例子配置了**iptables**，并且现在使用的是**SSH**进行连接的，那么会话恐怕已经被迫终止了！

为什么呢？因为我们已经把**OUTPUT**链策略更改为**DROP**了。此时虽然服务器能接收数据，但是无法发送数据：

```
# iptables -L
```

```
Chain INPUT (policy DROP)
```

```
target prot opt source destination
```

```
ACCEPT tcp -- anywhere anywhere tcp dpt:ssh
```

```
DROP all -- anywhere anywhere
```

```
Chain FORWARD (policy DROP)
```

```
target prot opt source destination
```

Chain OUTPUT (policy DROP)

target prot opt source destination

七、配置应用程序规则

尽管5.4节已经介绍了如何初步限制除SSH以外的其他连接，但是那是在链默认策略为ACCEPT的情况下实现的，并且没有对输出数据包进行限制。本节在上一节基础上，以SSH和HTTP所使用的端口为例，教大家如何在默认链策略为DROP的情况下，进行防火墙设置。在这里，我们将引进一种新的参数-m state，并检查数据包的状态字段。

1.SSH

1.允许接收远程主机的SSH请求

```
iptables -A INPUT -i eth0 -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
```

2.允许发送本地主机的SSH响应

```
iptables -A OUTPUT -o eth0 -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT
```

-m state: 启用状态匹配模块（state matching module）

--state: 状态匹配模块的参数。当SSH客户端第一个数据包到达服务器时，状态字段为NEW；建立连接后数据包的状态字段都是ESTABLISHED

--sport 22: sshd监听22端口，同时也通过该端口和客户端建立连接、传送数据。因此对于SSH服务器而言，源端口就是22

--dport 22: ssh客户端程序可以从本机的随机端口与SSH服务器的22端口建立连接。因此对于SSH客户端而言，目的端口就是22

如果服务器也需要使用SSH连接其他远程主机，则还需要增加以下配置：

1.送出的数据包目的端口为22

```
iptables -A OUTPUT -o eth0 -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
```

2.接收的数据包源端口为22

```
iptables -A INPUT -i eth0 -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT
```

2.HTTP

HTTP的配置与SSH类似：

1.允许接收远程主机的HTTP请求

```
iptables -A INPUT -i eth0 -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
```

1.允许发送本地主机的HTTP响应

```
iptables -A OUTPUT -o eth0 -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT
```

3.完整的配置

1.删除现有规则

```
iptables -F
```

2.配置默认链策略

```
iptables -P INPUT DROP
```

```
iptables -P FORWARD DROP
```

```
iptables -P OUTPUT DROP
```

3.允许远程主机进行SSH连接

```
iptables -A INPUT -i eth0 -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
```

```
iptables -A OUTPUT -o eth0 -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT
```

4.允许本地主机进行SSH连接

```
iptables -A OUTPUT -o eth0 -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
```

```
iptables -A INPUT -i eth0 -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT
```

5.允许HTTP请求

```
iptables -A INPUT -i eth0 -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
```

```
iptables -A OUTPUT -o eth0 -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT
```

上一篇 [IPtables中SNAT、DNAT和MASQUERADE的含义](#)

下一篇 [awk用法详解](#)

相关文章推荐

- [iptables基础知识.详解](#)
- [iptables的基础知识-iptables中的状态检测](#)
- [Cisco路由技术基础知识详解](#)
- [Android基础知识详解【PDF文档】](#)
- [Linux操作系统文件系统基础知识详解](#)
- [java中的面向对象基础知识详解](#)
- [【Java基础知识 - 08】UML详解](#)
- [【Java基础知识 - 05】OOAD详解](#)
- [iptables基础知识](#)
- [Spring基础知识详解](#)

猜你在找

[机器学习之概率与统计推断](#)

[机器学习之凸优化](#)

[响应式布局全新探索](#)

[深度学习基础与TensorFlow实践](#)

[前端开发在线峰会](#)

[机器学习之数学基础](#)

[机器学习之矩阵](#)

[探究Linux的总线、设备、驱动模型](#)

[深度学习之神经网络原理与实战技巧](#)

[TensorFlow实战进阶：手把手教你做图像识别应用](#)

[查看评论](#)

1楼 [zhushuangyin](#) 2017-03-30 10:12发表



我尽量以通俗易懂的方式总结了iptables的概念，欢迎交流

<http://www.zsythink.net/archives/1199>

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

[公司简介](#) | [招贤纳士](#) | [广告服务](#) | [联系方式](#) | [版权声明](#) | [法律顾问](#) | [问题报告](#) | [合作伙伴](#) | [论坛反馈](#)

[网站客服](#)

[杂志客服](#)

[微博客服](#)

webmaster@csdn.net

400-660-0108 |

北京创新乐知信息技术有限公司 版权所有 | [江苏知之为计算机有限公司](#) |

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved

