第 12 讲　防火墙与代理服务器

王晓庆

wangxiaoqing@outlook.com

June 20, 2016

Outline

将内网接入 Internet 的主要方式

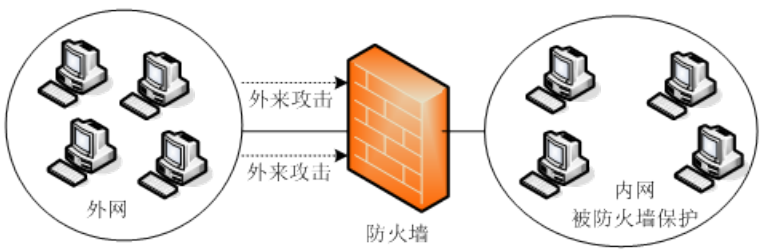
- 路由器
- 防火墙
- 代理服务器
- NAT 网关
- VPN

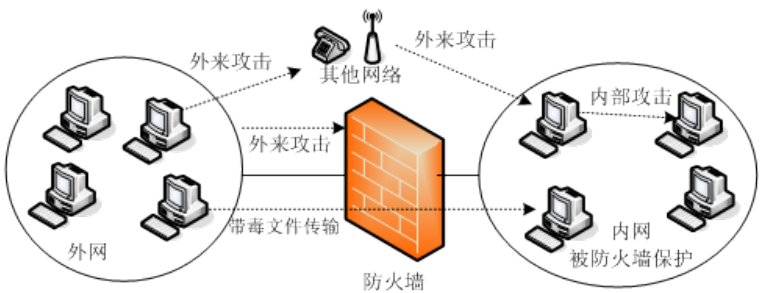
说明

实际应用中，很少有使用单一方式的，往往是组合或集成上述多种方式，例如将防火墙、代理服务器和 NAT 进行集成。

防火墙的作用

- 在内外网之间安装防火墙，形成一个保护层
- 对进出的所有数据进行监测、分析、限制，并对用户进行认证，防止有害信息进入受保护的网络
- 存在局限性，如不能防范绕过防火墙的攻击；不能防止受到病毒感染的软件或文件的传输，以及木马攻击等；难以避免来自内部的攻击

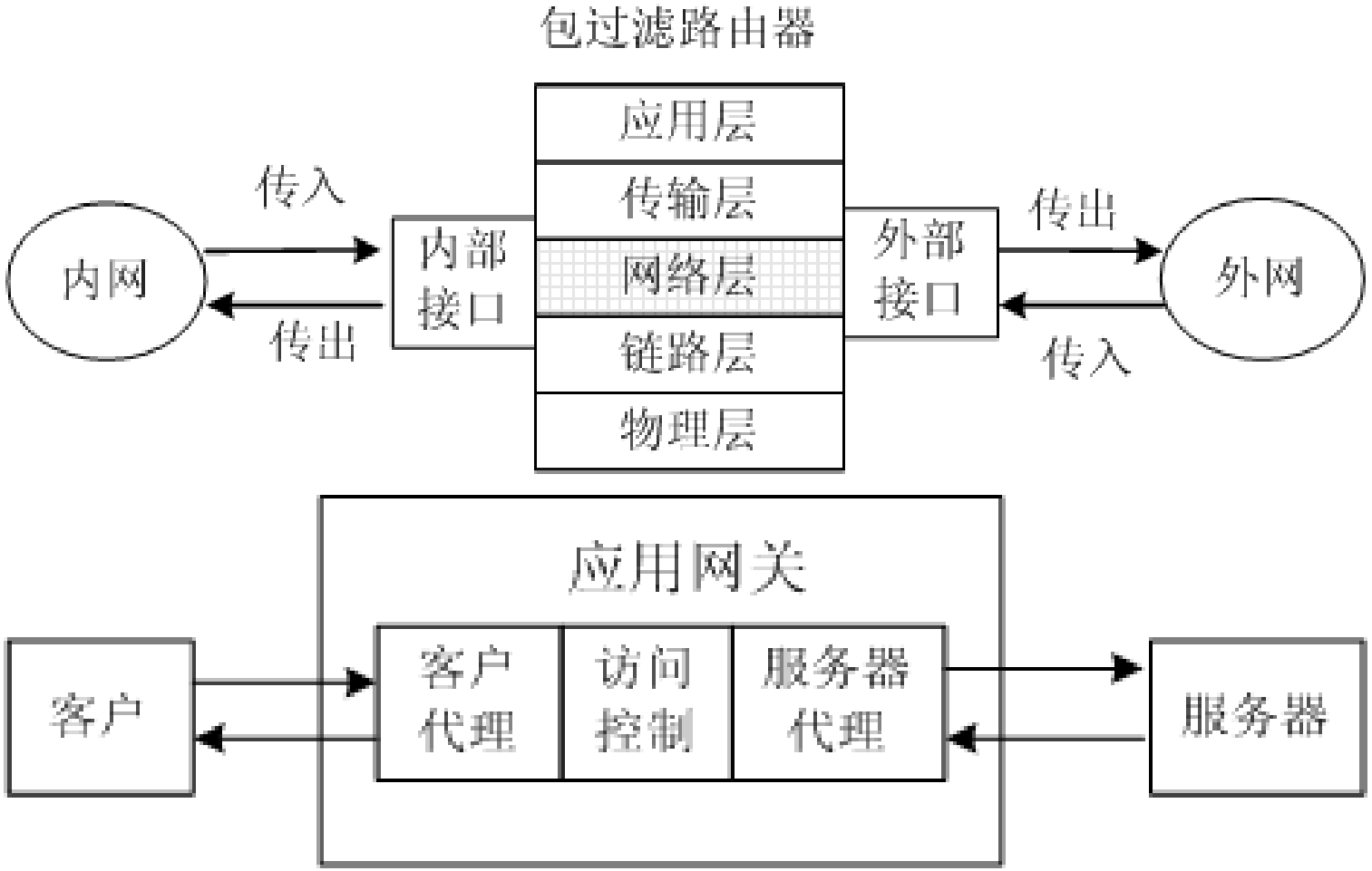




防火墙的类型 (1)

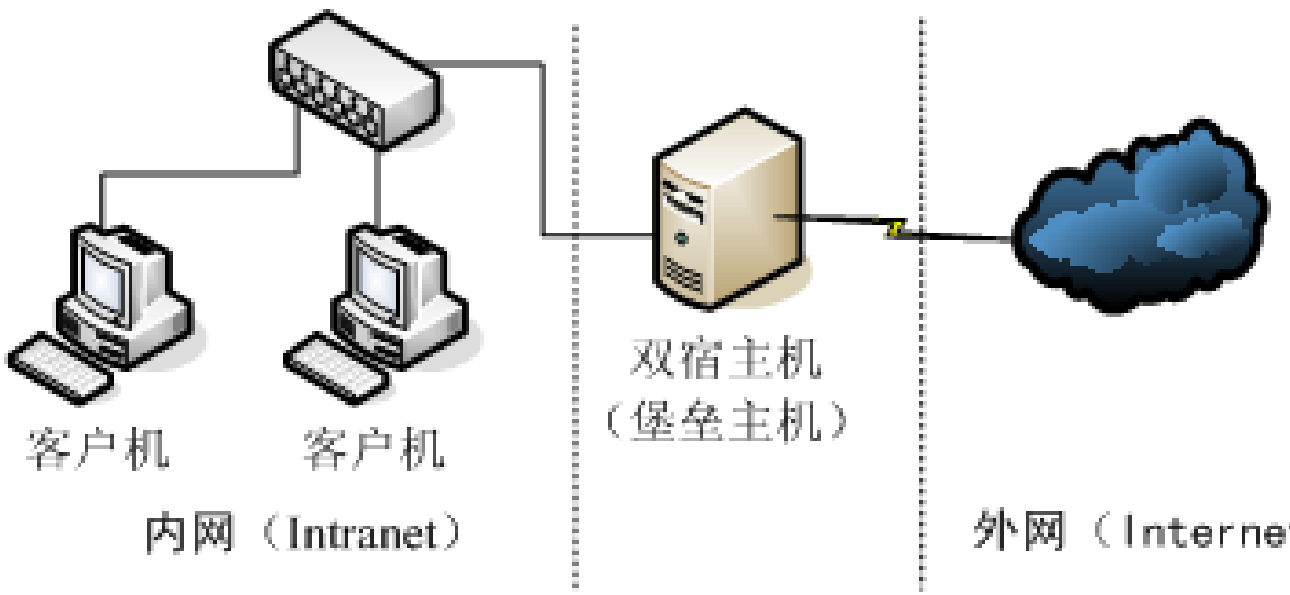
- 包过滤 (Packet Filtering) 路由器
  - 在网络层对数据包进行选择，选择的依据是设置的过滤规则，通过检查数据流中每个数据包的源地址、目的地址、所用的端口号、协议状态等因素，确定是否允许该数据包通过
- 应用网关
  - 工作在网络体系结构的应用层，又称代理服务器，是应用级防火墙。应用网关采用代理技术提交请求和应答，不给内外网计算机直接会话机会，优点是安全，缺点是速度相对较慢。
- 状态检测防火墙
  - 在检查数据包的基础上，也检查连接状态和应用状态信息，利用数据包之间的关联信息来避免不必要的包检查。更高级的状态检测还能基于应用状态来过滤通信

防火墙的类型 (2)


包过滤路由器
应用层
传输层
网络层
链路层
物理层
传入
传出
内网
内部接口
外部接口
传出
传入
外网
应用网关
客户
客户代理
访问控制
服务器代理
服务器

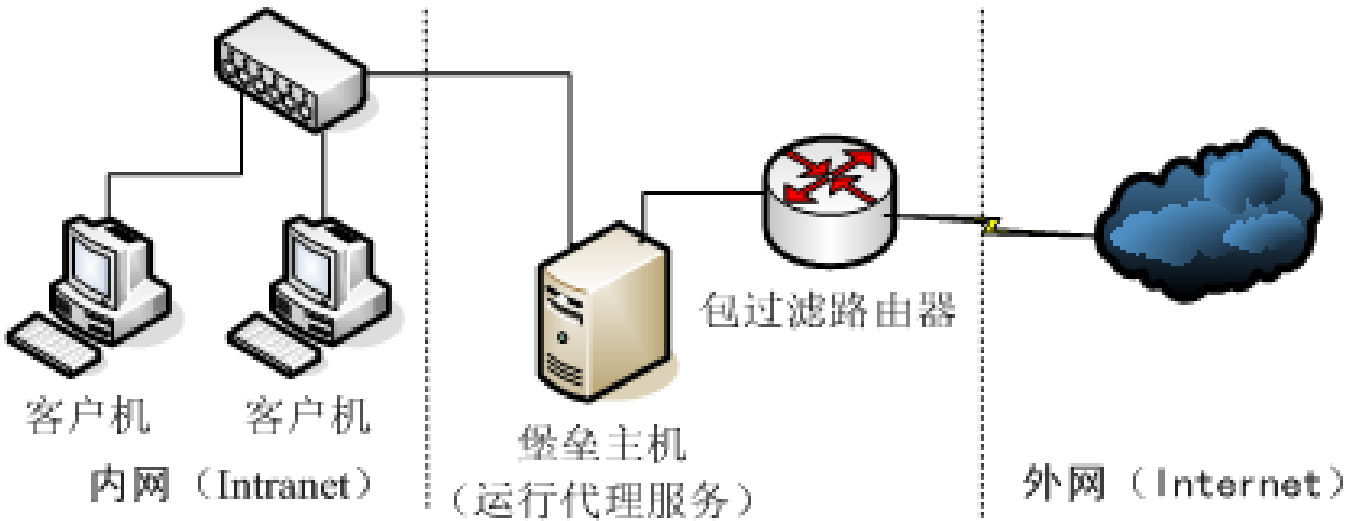
防火墙配置方案 (1)

- 双宿主机网关 (Dual Homed Gateway)
  - 用一台配有两个网络接口的双宿主机做防火墙, 其中一个网络接口连接内网 (被保护网络), 另一个连接 Internet。双宿主机又称堡垒主机, 用于运行防火墙软件



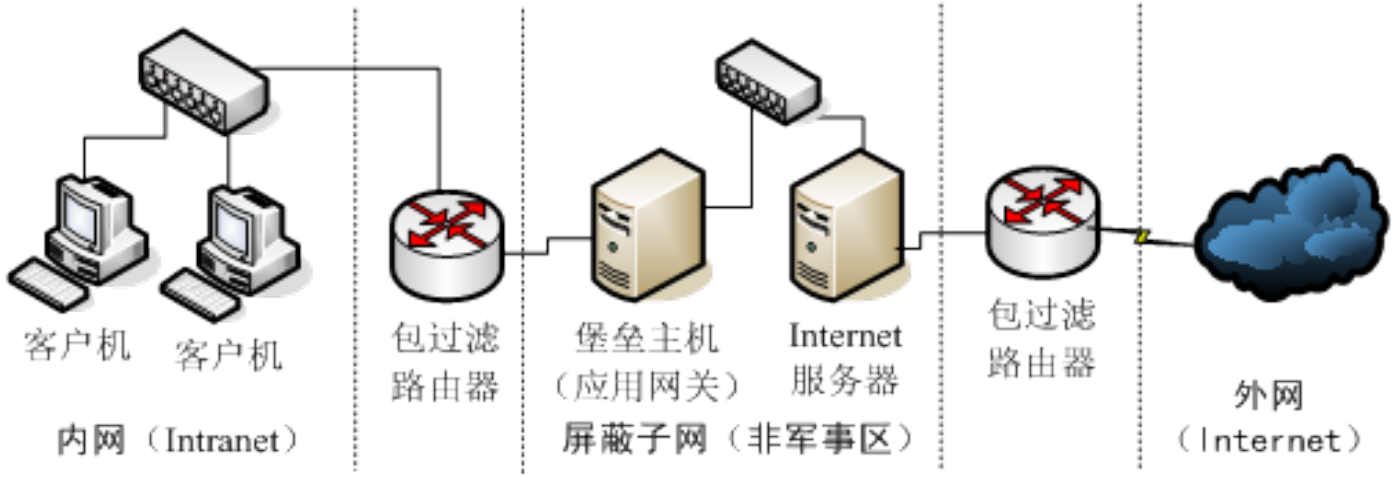
防火墙配置方案 (2)

- 屏蔽主机网关 (Screened Host Gateway)
  - 可分为单宿型和双宿型两种类型。通常采用双宿型，堡垒主机有两块网卡，一块连接内网，一块连接包过滤路由器，双宿堡垒主机在应用层提供代理服务



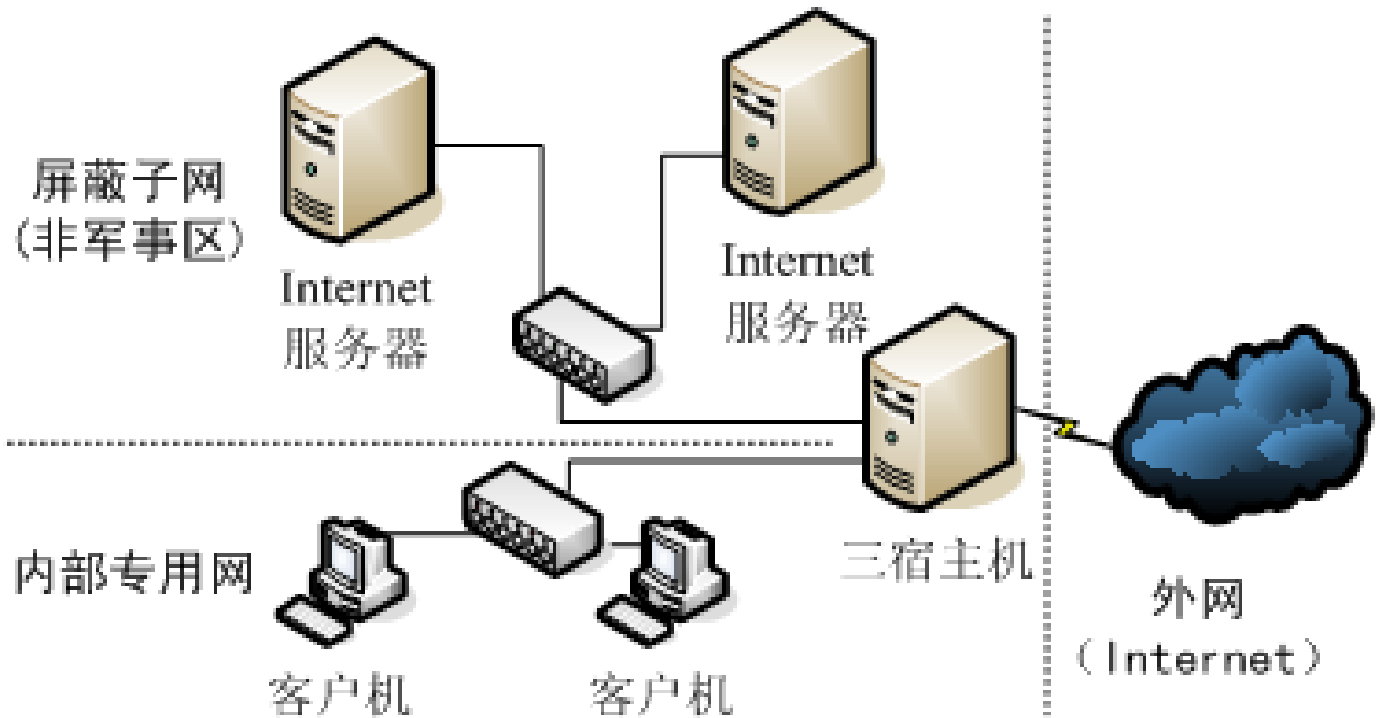
防火墙配置方案 (3)

- 屏蔽子网 (Screened Subnet)
  - 最为复杂的防火墙体系，在内网和 Internet 之间建立一个被隔离的子网，该子网与内网隔离，形成一个网络防御带，在其中安装应用服务器以发布公共服务。屏蔽子网又称周边网络或 DMZ。
  - 多防火墙屏蔽子网
    - 最典型的是用两个包过滤路由器将屏蔽子网分别与内网和 Internet 隔开，构成一个“缓冲地带”



防火墙配置方案 (4)

- 三宿主机屏蔽子网
  - 一台防火墙主机共有 3 个网络接口，分别连接到内部专用网、屏蔽网络和外网 (Internet)



Linux 防火墙解决方案

- Linux 提供优秀的防火墙软件 Netfilter/iptables，可以在一台低配置的计算机上运行，以替代昂贵的硬件防火墙产品
- 就功能特性来说，可以将防火墙分为以下 3 种类型：
  1. NAT：让内网通过一个或多个公网 IP 地址访问公网，作为一种防火墙技术，将内网 IP 地址隐藏起来使公网用户无法直接访问内网。
  2. 包过滤：依据过滤规则读取和处理网关的所有数据包，允许或阻止数据包通过网关，是一种最基本的防火墙技术。
  3. 代理服务器：代表内网主机与外部主机通信，通常是应用级网关。作为防火墙技术，隔离内外网，并提供访问控制和网络监控功能。

Linux 的 NAT 技术

- Linux 的 Netfilter/iptables 支持源 NAT 和目的 NAT。
- NAT 对数据包的源 IP 地址、目的 IP 地址、源端口、目的端口进行改写，据此将 NAT 分为以下两种类型：

1. 源 NAT(SNAT)

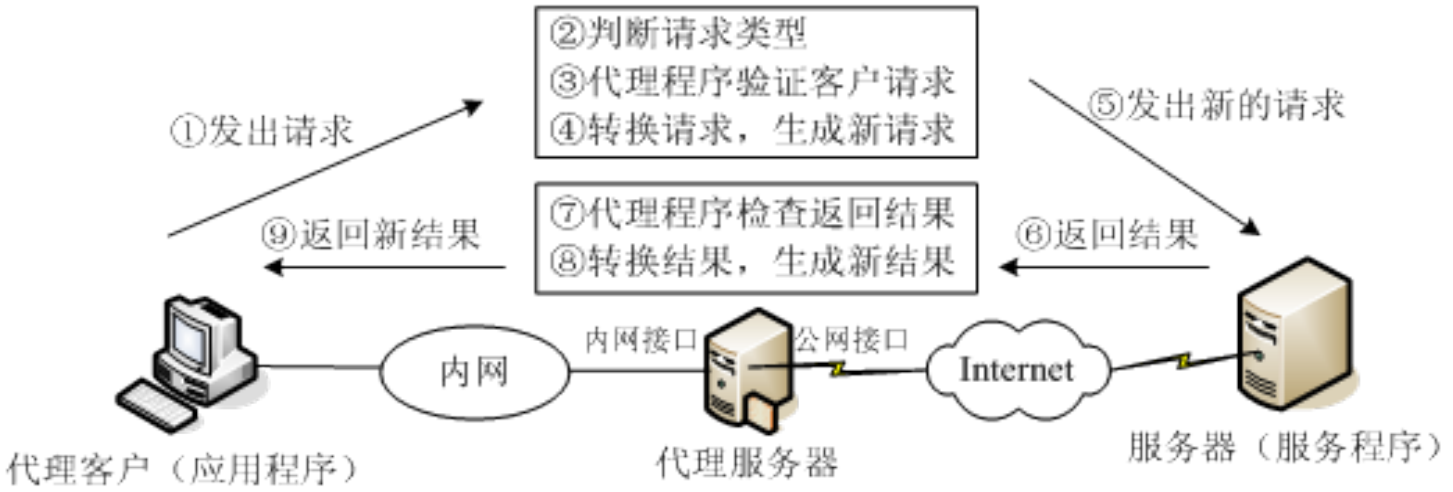
改变数据包的源地址。网络连接共享属于源 NAT,IP 伪装 (IP Masquerade) 是源 NAT 的一种特殊形式。

1. 目的 NAT(DNAT)

改变数据包的目的地址，它与源 NAT 相反。例如，端口转发、负载均衡和透明代理就是属于目的 NAT。

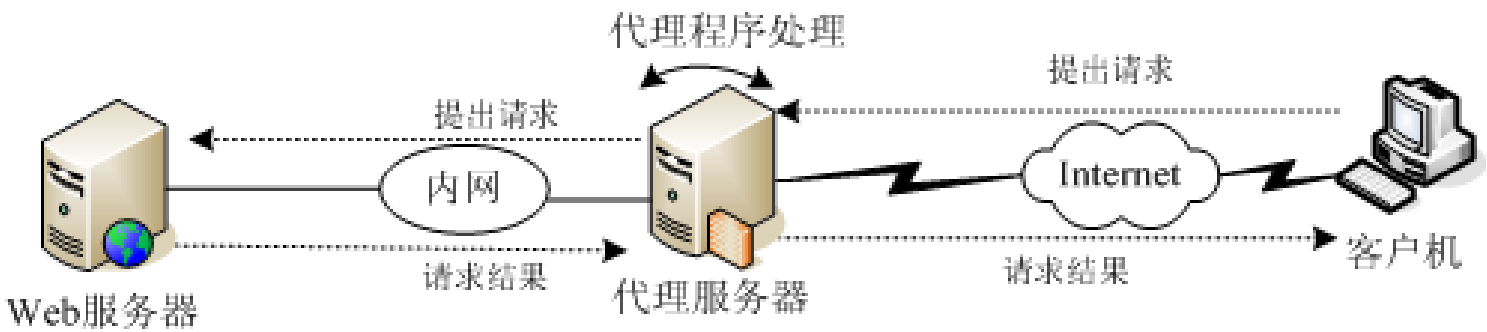
代理服务器技术 (1)

- 代理服务器工作原理
  - 应用层代理是最典型的代理方式，狭义的代理服务往往指这种方式
  - 在客户端和服务器之间建立连接并转发数据
  - 工作在应用层，多数代理服务器只支持部分应用程序，一般支持 HTTP 代理
  - 复杂的应用层代理还能够缓存、过滤和优化数据
  - 代理服务器至少有两个网络接口，一个连接内网，另一个连接 Internet



代理服务器技术 (2)

- 反向代理技术
  代理服务器也可为外网用户访问内网提供代理服务，通常将这种代理服务称为反向代理或逆向代理
  - 通常只用来发布内网 Web 服务器
  - 不仅充当防火墙以防止外网用户直接与 Web 服务器通信，还可充当 Web 缓冲服务器，以降低实际 Web 服务器负载，提高访问速度
  - 可对用户身份进行认证，对访问内容进行过滤
  - 常用于网络负载均衡和故障热处理，对性能要求很高。

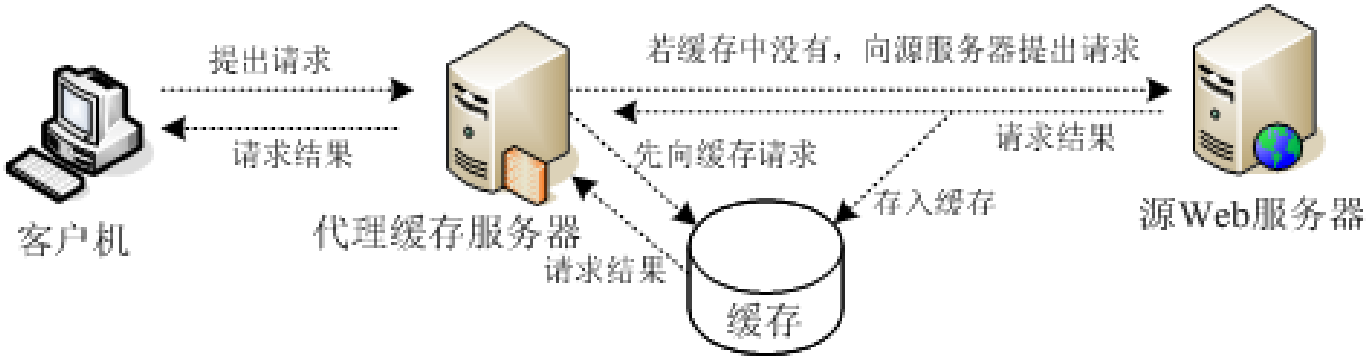


代理服务器技术 (3)

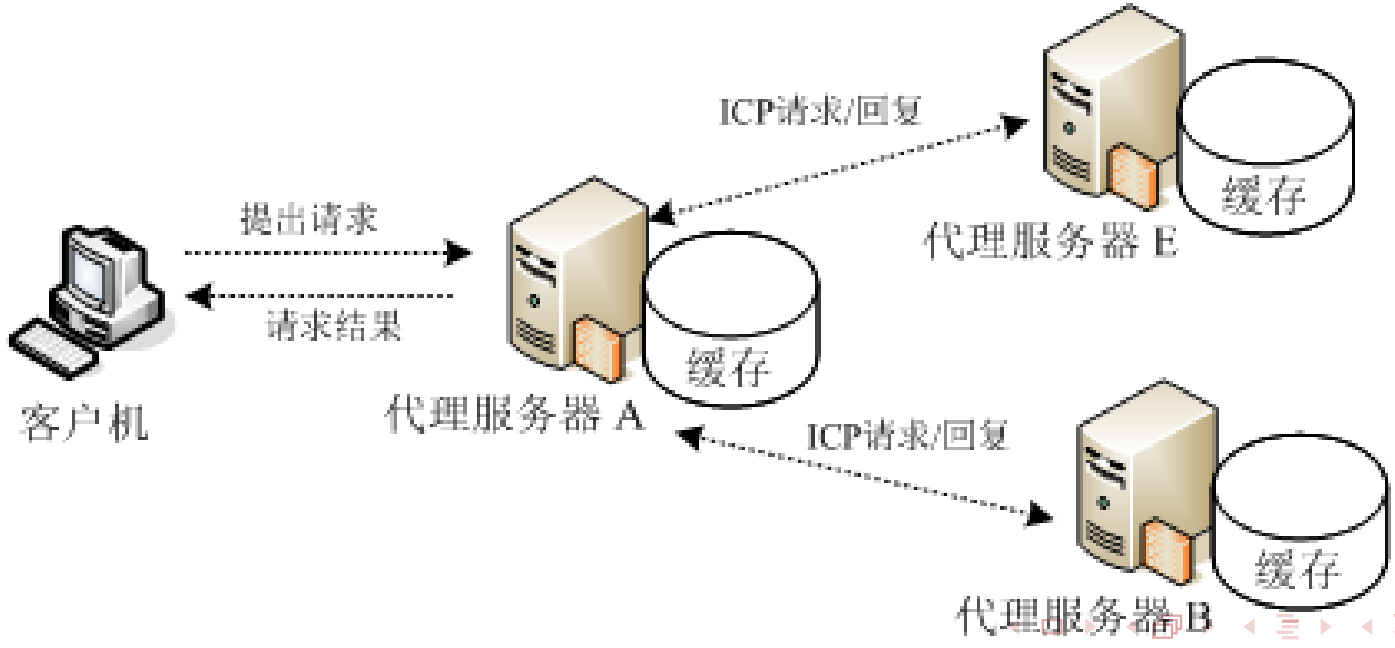
- 缓存 (代理缓存服务器)
  - 缓存由一个或多个分区组成, 此处分区是指磁盘上用作缓存的存储区域
  - 当使用代理缓存时, 用户的 Web 请求被发送到代理服务器
  - 代理服务器首先请求缓存中的 Web 信息, 如果缓存中没有, 就向源 Web 服务器请求信息并将其存入缓存中, 然后再发送信息给请求的用户
- 缓存方案主要有 3 种类型
  1. 标准代理缓存 (正向代理)
  2. HTTP(Web) 加速器 (即反向代理)
  3. ICP(Internet 缓存协议) 多层缓存

代理服务器技术 (4)

- 标准代理缓存

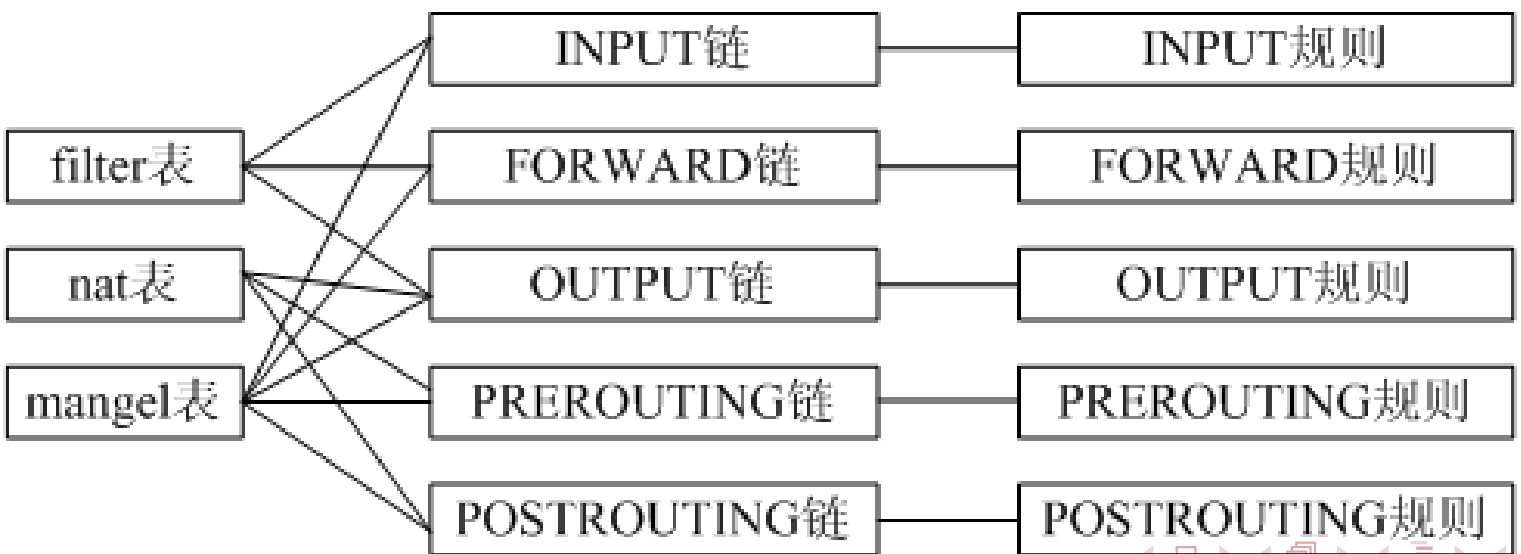


- ICP 多层缓存



Netfilter 架构 (1)

- 概述
  - Netfilter 位于网络层与防火墙内核之间，是 Linux 内核中的一个通用架构，定义了包过滤子系统功能的实现。
  - iptables 使用 Netfilter 架构在 Linux 内核中管理包过滤
  - Netfilter 提供 3 个表 (tables)，每个表由若干个链 (chains) 组成，而每条链可以由若干条规则 (rules) 组成
  - 可以将 Netfilter 看成是表的容器，将表看成是链的容器，将链看成是规则的容器
  - 表是所有规则的总和，链是在某一检查点上的规则的集合。



Netfilter 架构 (2)

- 表
  - filter(过滤网络数据包)
  - nat(修改数据包来创建新的连接，实现网络地址转换)
  - mangle(处理特定的数据包)

Netfilter 架构 (3)

- 链
  - filter 表内置链
    - INPUT(处理目标地址是本机的网络数据包，即检测过滤传入数据包)
    - FORWARD(处理经本机转发的数据包，即检测过滤路由数据包)
    - OUTPUT(处理由本地产生要发送的网络数据包，即检测过滤传出数据包)
  - nat 表内置链
    - PREROUTING(包含路由前的规则，转换需要转发数据包的目的地址)
    - POSTROUTTNG(包含路由后的规则，转换需要转发数据包的源地址)
    - OUTPUT(转换本地数据包的目的地址)
  - mangle 表内置链
    - INPUT、OUTPUT、FORWARD、PRFROUTTNG 和 POSTROUTTNG

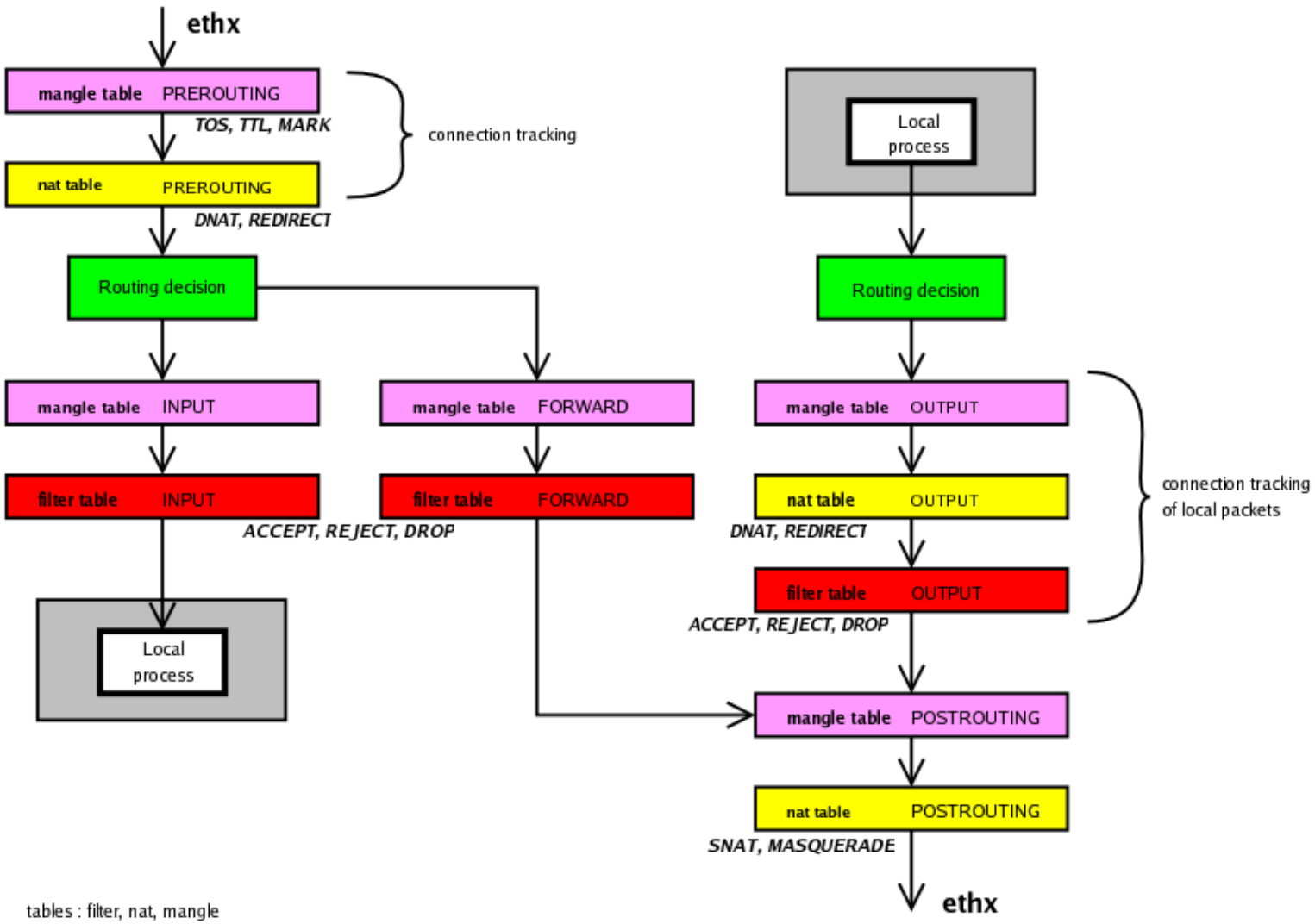
Netfilter 架构 (4)

- 规则
  - 每一条链中可以有一条或多条规则
  - 每条规则定义所要检查的数据包的特征或条件，如源地址、目的地址、传输协议等，以及处理匹配条件的包的方法，如允许、拒绝等
  - 当一个数据包到达一个链时,iptables 从链中第 1 条规则开始检查，判断该数据包是否满足规则所定义的条件，如果满足就按照所定义的方法处理该数据包；否则继续检查下一条规则，如果不符合链中任何规则,iptables 根据该链预定义的默认策略来处理数据包

Netfilter 架构 (5)

- 包处理流程
  - 数据包到达 Linux 网络接口，根据定义的规则进行处理
  - 涉及 Netfilter 的 3 个表 (filter、nat 和 mangle)，每个表又有不同的链
  - filter 表用于实现防火墙功能，内置的 3 个链 INPUT、FORWAR 和 OUTPUT，分别对包的传入、转发和传出进行过滤处理
  - nat 表用于实现地址转换和端口转发功能，内置的 3 个链 PREROUTING、POSTROUTING 和 OUTPUT，分别对转发数据包目的地址、转发数据包源地址和本地数据包目的地址进行转换。
  - mangle 表则是一个自定义表，用于各种自定义操作，而且 mangle 表中的链在 Netfilter 包处理流程中处于优先的位置。实际应用中很少用到 mangle 表。

Netfilter 架构 (6)

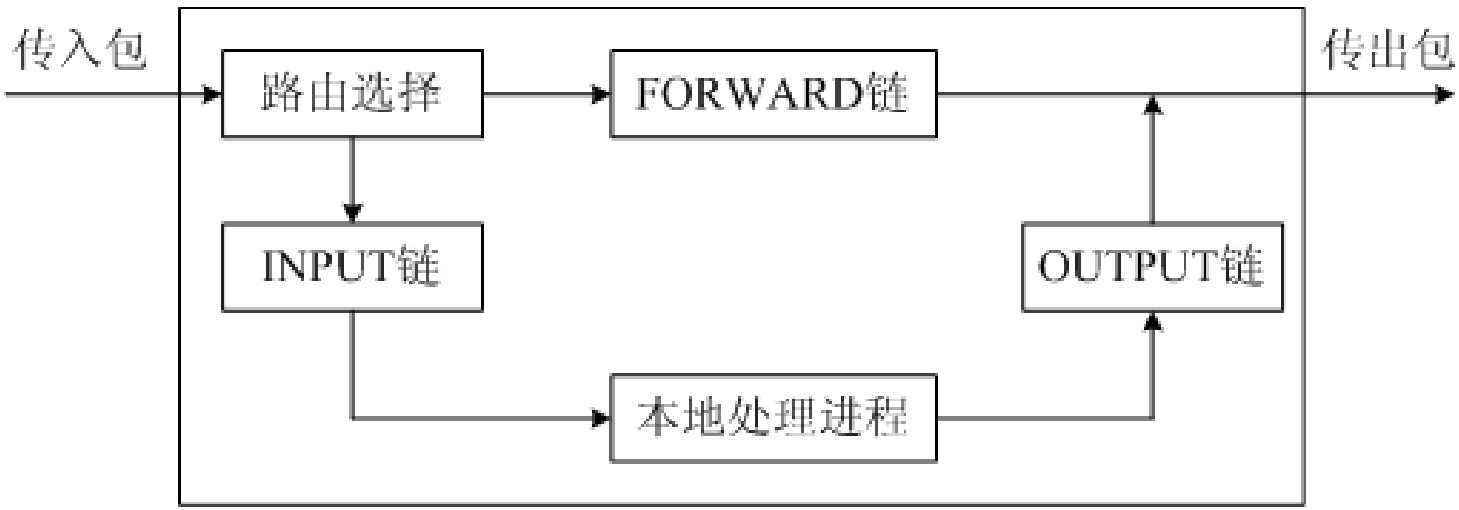


tables : filter, nat, mangle
chains: PREROUTING, INPUT, FORWARD, OUTPUT, POSTROUTING
targets: ACCEPT, DNAT, DROP, LOG, MARK, MASQUERADE, MIRROR, QUEUE, REDIRECT, REJECT, RETURN, SNAT, TOS, TTL, ULOG, ...

包过滤机制 (1)

- filter 表用于实现包的过滤处理，内置 3 个链：
  - INPUT 链过滤从内网或外网发往防火墙本身的数据包
  - OUTPUT 链过滤从防火墙本身发往内网或外网的数据包
  - FORWARD 链过滤内外网之间通过防火墙转发的数据包
  - 除了 3 个内置链之外，管理员可根据需要添加自定义的链
- 当数据包到达防火墙时,Linux 内核首先根据路由表决定数据包的目标，若数据包的目的地址是本机，则将数据包送往 INPUT 链进行规则检查；若目的地址不是本机，则检查内核是否允许转发，如果允许，则将数据包送往 FORWARD 链进行规则检查，如果不允许转发，则丢弃该数据包。对于防火墙主机本地进程产生并准备发出的数据包，则交由 OUTPUT 链进行规则检查。

包过滤机制 (2)



包过滤机制 (3)

- 包通信方向
  - 应正确理解每个接口上数据通信的方向
  - 同一数据包通过不同的网络接口，通信方向不同
  - 从内网到外网的通信，在内网接口上为传入通信，在外网接口上为传出通信
  - 从外网到内网的通信，在外网接口上为传入通信，在内网接口上为传出通信

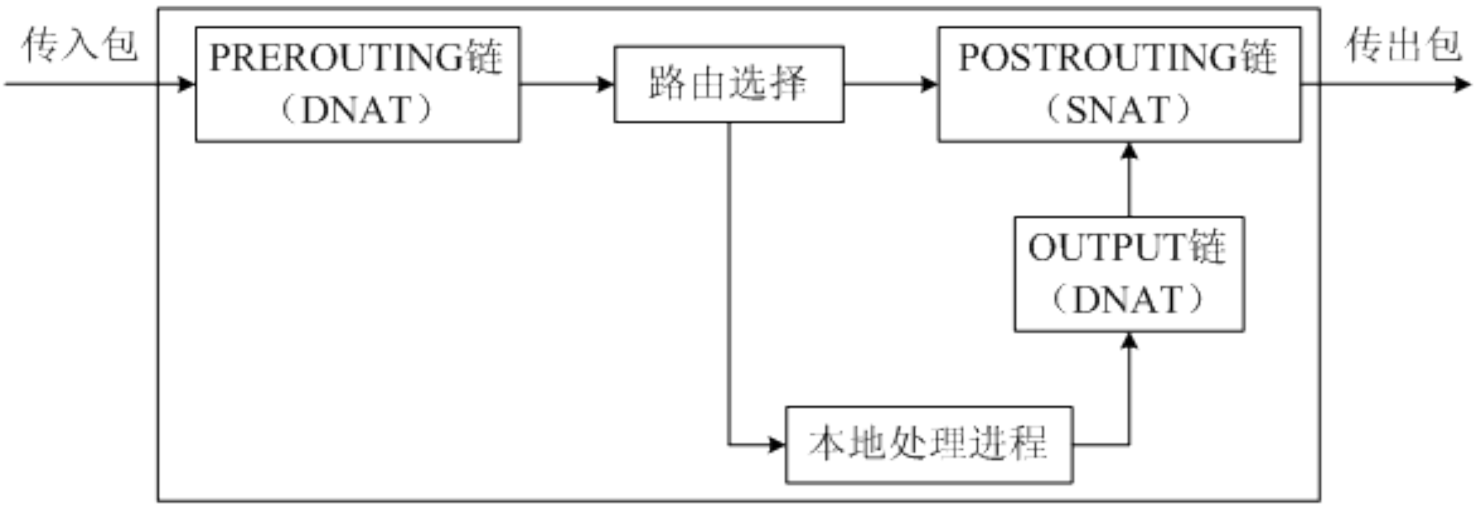
网络地址转换机制 (1)

- 网络地址转换类型
  - 数据包的源地址 (或端口) 或目的地址 (或端口) 修改需要通过 nat 表来实现
  - 对源地址或源端口进行替换修改，称为 SNAT(源 NAT)
  - 对目的地址或端口进行替换修改，称为 DNAT(目的 NAT)

网络地址转换机制 (2)

- nat 链
  - INPUT 链过滤从内网或外网发往防火墙本身的数据包
  - OUTPUT 链过滤从防火墙本身发往内网或外网的数据包
  - FORWARD 链过滤内外网之间通过防火墙转发的数据包
- 网络地址转换过程
  - 当数据包到达防火墙时，在还没有交给路由选择之前由 PREROUTING 链进行检查处理，该链可以对需要转发的数据包的目的地址和端口进行转换修改 (DNAT)，从而实现端口或主机重定向
  - 经过路由选择之后，所有要传出的包在 POSTROUTING 链中进行检查处理，该链可以对包的源地址或端口进行转换修改 (SNAT)。
  - 本地进程产生并准备传出的包则由 OUTPUT 链进行检查处理，该链也可进行 DNAT 操作。

网络地址转换机制 (3)


传入包
PREROUTING链
（DNAT）
路由选择
POSTROUTING链
（SNAT）
传出包
OUTPUT链
（DNAT）
本地处理进程

iptables 命令组成

- iptables 命令格式

```
iptables [-t table] cmd chain [options]
```

- 说明
  - table: 指定这个规则所应用的规则表 (filter, nat, mangle)，如果没有使用这个选项, 默认指定 filter 表
  - cmd: 指定要执行的动作，如添加或删除一条规则
  - chain: 指定编辑、创建或删除的链
  - options: 选项，如匹配规则和/或动作

iptables 命令 (1)

-A(–append)

```
iptables -A INPUT -j ACCEPT
# 向 filter 表的 INPUT 链追加一条规则：
# 接受所有目标地址为本机的数据包
# 新增加的规则将会成为规则链中的最后一条规则
```

-D(–delete)

```
iptables -D INPUT -p tcp --dport 80 -j DROP
# 从 filter 表中删除规则：
# 拒绝协议为 tcp，目标地址为本机 80 端口的数据包
# 也可以通过指定规则编号加以删除，如：
iptables -D INPUT 1 # 删除 INPUT 链中编号为 1 的规则
```

iptables 命令 (2)

-I(–insert)

```
iptables -I INPUT 1 -p tcp --dport 80 -j ACCEPT
# 在 filter 表的 INPUT 链中位置 1 处插入一条规则：
# 允许协议为 tcp，目标地址为本机 80 端口的数据包
# 原来在位置 1 及其后的规则依次向后移动
```

–L(–list)

```
iptables -L INPUT
# 列出 filter 表的 INPUT 链中的所有规则
iptables -L # 列出 filter 表的所有规则
iptables -t nat -L # 列出 nat 表的所有规则
```

iptables 命令 (3)

-F(–flush)

```
iptables -F INPUT
# 清空 filter 表的 INPUT 链中的所有规则
iptables -t nat -F
# 清空 nat 表的所有规则
```

-Z(–zero)

```
iptables -Z INPUT
# 将 filter 表的 INPUT 链的计数器清零
iptables -Z
# 将 filter 表的所有链的计数器清零
```

iptables 命令 (4)

-R(–replace)

```
iptables -R INPUT 1 -p tcp --dport 80 -j DROP
# 替换 filter 表的 INPUT 链的规则 1：
# 丢弃目标地址为本机 tcp 协议 80 端口的数据包
```

-P(–policy)

```
iptables -P INPUT DROP
# 设置 filter 表的 INPUT 链的默认处理策略为 DROP
# 当数据包经过 INPUT 链且没有匹配任何规则时将被丢弃
# 只有内置链才能定义默认处理策略
```

iptables 命令 (5)

-N(–new-chain)

```
iptables -N MYCHAIN
# 在 filter 表中定义一条名为 MYCHAIN 的新链
# 定义了新链后，可以在新链中添加规则
```

-X(–delete-chain)

```
iptables -X MYCHAIN
# 删除 filter 表中名为 MYCHAIN 的链
# 只能删除用户自定义的链，不能删除内置链
# 被删除的自定义链必须没有被引用
```

-E(–rename-chain)

```
iptables -E MYCHAIN MYNEWCHAIN
# 将用户自定义的链 MYCHAIN 重命名为 MYNEWCHAIN
```

iptables 匹配参数 (1)

-p(–protocol)

```
iptables -A INPUT -p tcp -j ACCEPT
#-p 可以匹配协议类型 tcp,udp,icmp 或 all
#-p ! tcp 表示匹配 tcp 之外的其他协议
```

-s(–source)

```
iptables -A INPUT -s 192.168.1.1 -j ACCEPT
# 允许源地址为 192.168.1.1 且目的地址为本机的数据包
#-s 192.168.1.0/24 匹配一个网段的源地址
#-s ! 192.168.2.0/24 排除一个网段的源地址
```

iptables 参数 (2)

-d(–destination)

```
iptables -A OUTPUT -d 192.168.1.1 -j ACCEPT
# 允许源地址为本机且目的地址为 192.168.1.1 的数据包
```

-j(–jump)

```
# 定义规则的目标，即对匹配该规则的数据包所做的操作。
# 目标可以是用户自定义链
# 目标也可以是一个内置目标（ACCEPT,DROP 等）或扩展
```

-g(–goto)

```
# 设置数据包继续到用户自定义链进行处理
```

iptables 参数 (3)

-i(–in-interface)

```
iptables -A INPUT -i eth0 -j ACCEPT
# 允许从 eth0 接口进入且目的地值为本机的数据包
```

-o(–out-interface)

```
iptables -A FORWARD -o eth1 -j ACCEPT
# 允许源地址和目的地址都不是本机从 eth1 接口转发出去
```

iptables 参数 (4)

-f(–fragment)

```
# 设置只对分段（分片）的数据包应用当前规则
```

-c(–set-counters) PKTS BYTES

```
# 为指定规则重设（初始化）计数器，可指定重设的包计数器或字节计
```

iptables 匹配扩展 (1)

- 不同的网络协议提供不同的匹配扩展，前提是该协议必须现在 iptables 指令中指定。
- -p tcp 的匹配扩展

–sport(–source-port)

```
iptables -A INPUT -p tcp --sport 1:1024 -j ACCEPT
# 允许源端口号为 1-1024 且目的地址为本机的 tcp 包
```

–dport(–destination-port)

```
iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
# 允许从本机发出的目标端口号为 80 的 tcp 包
```

–syn

```
--syn 匹配 syn 包，! --syn 匹配非 syn 包
```

iptables 匹配扩展 (2)

- -p tcp 的匹配扩展 (续)

–tcp-flags

```
iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST SYN
# 检查数据包的 SYN, ACK, FIN, RST 位
# 仅匹配设置了 SYN 位, 而未设置 ACK,FIN,RST 位的数据包
```

–tcp-options

```
iptables -A FORWARD -p tcp --tcp-options 6
# 检查 tcp 选项类型, 匹配选项类型为 6 的数据包
# 选项类型值可参考
http://www.iana.org/assignments/
tcp-parameters/tcp-parameters.xhtml
```

iptables 匹配扩展 (2)

- -p udp 的匹配扩展
  - –sport
  - –dport
- -p icmp 的匹配扩展

–icmp-type

```
iptables -A INPUT -p icmp --icmp-type 8 -j ACCEPT
# 允许目标地址为本机且类型值为 8(echo request) 的 icmp 包
# 可以写类型值或类型名
iptables -p icmp -h # 可查看 icmp 类型列表
```

iptables 匹配扩展 (3)

-m limit 的匹配扩展

```
iptables -A INPUT -m limit --limit 100/s \
--limit-burst 120 -j ACCEPT
# 突发收到 120 个数据包后立即触发
# 每秒仅允许通过 100 个数据包的限制
#--limit 的时间单位可以是 s(秒),m(分),h(时),d(日)
```

-m mac 的匹配扩展

```
iptables -A FORWARD -m mac \
--mac-source 00:50:0C:34:9A:D3 -j DROP
# 丢弃来自指定 MAC 地址发出的数据包
# 注意：一个数据包经过路由器转发后,
#       其源 MAC 地址将变成路由器接口的 MAC 地址！
```

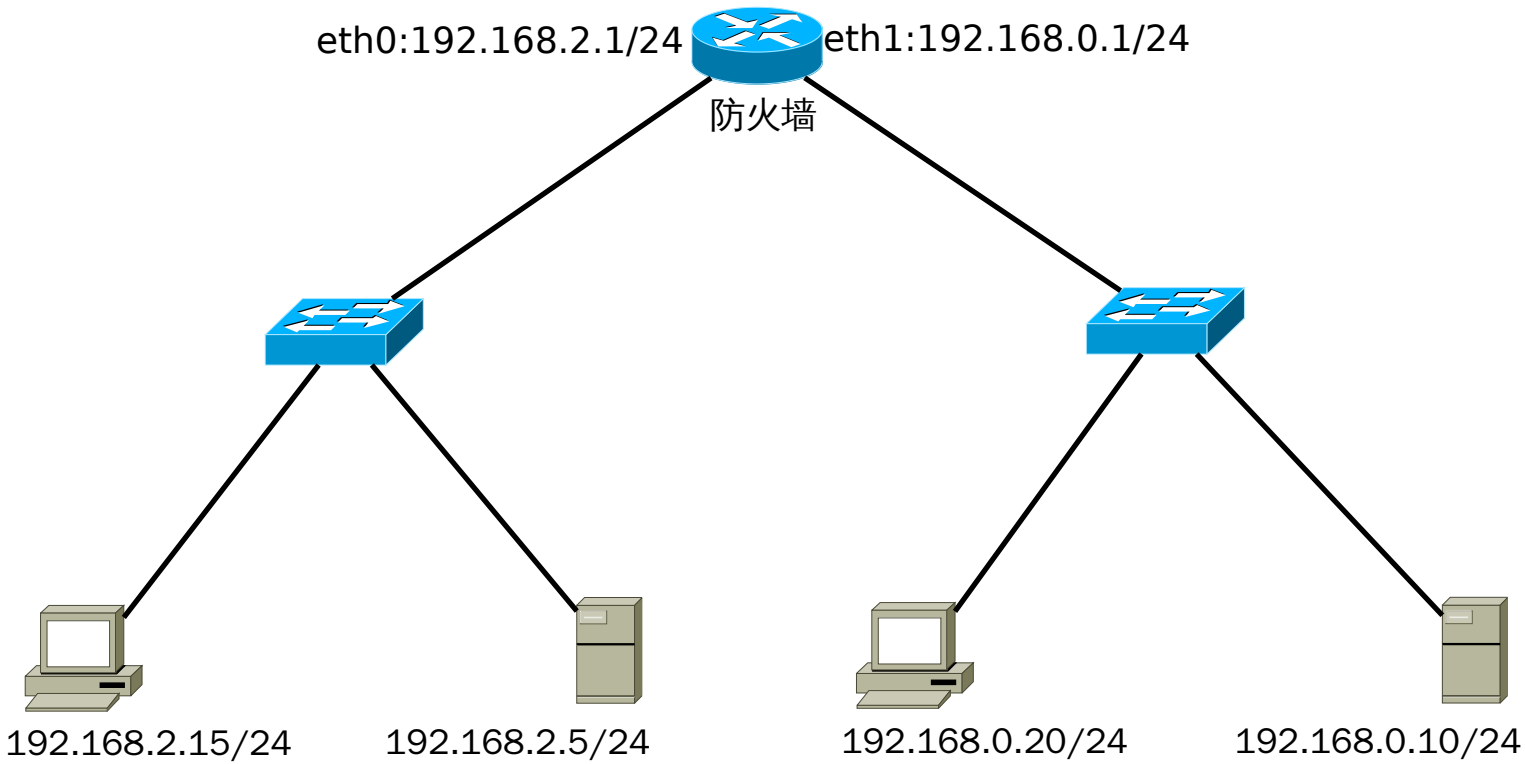
iptables 目标选项 (1)

- 当一个数据包与一个特定的规则相匹配时，这条规则可以将这个数据包重新定向到不同的目标中，由这些目标来决定对这个数据包如何处理。目标选项通过-j 参数指定，可分为 3 种类型
- 标准的目标选项
  - ACCEP: 允许数据包通过并发送到它的目的地或其他链。
  - DROP: 丢弃数据包，不给发送者任何反馈信息。
  - QUEUE: 将数据包放置在一个用户空间应用程序的队列中等待被处理。
  - RETURN: 停止遍历当前链中的规则，恢复到先前链的下一条规则。

iptables 目标选项 (2)

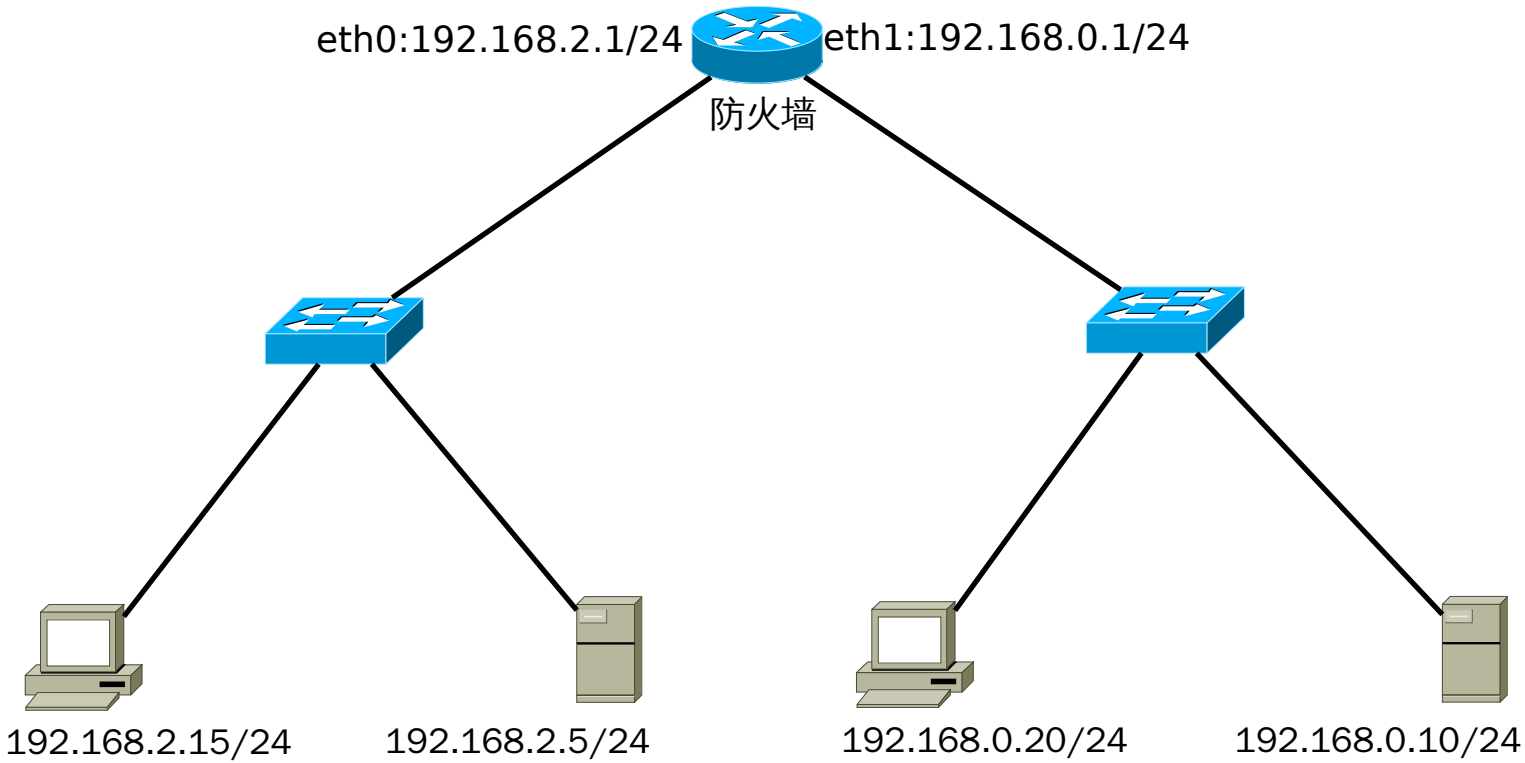
- 扩展的目标模块
  - LOG: 在日志中记录所有匹配这条规则的数据包
  - REJECT: 丢弃数据包, 并向远程系统发回一个错误通知
  - MASQUERADE: 将数据包来源 IP 转换为输出数据包的接口的 IP 以实现 IP 伪装。
  - SNAT: 将数据包来源 IP 和端口转换为某指定的 IP 和端口。
  - DNAT: 将数据包目的 IP 和端口转换为某指定的 IP 和端口。
  - REDIRECT: 将数据包重新转向到本机或另一台主机的某个端口, 通常用此功能实现透明代理或对外开放内网, 它相当于 DNAT 的一个特例。
- 自定义链
  - 规则的目标也可以是一个自定义链的名称, 该链应事先建立好, 并在链中设置好相应的规则。

iptables 命令练习 (1)



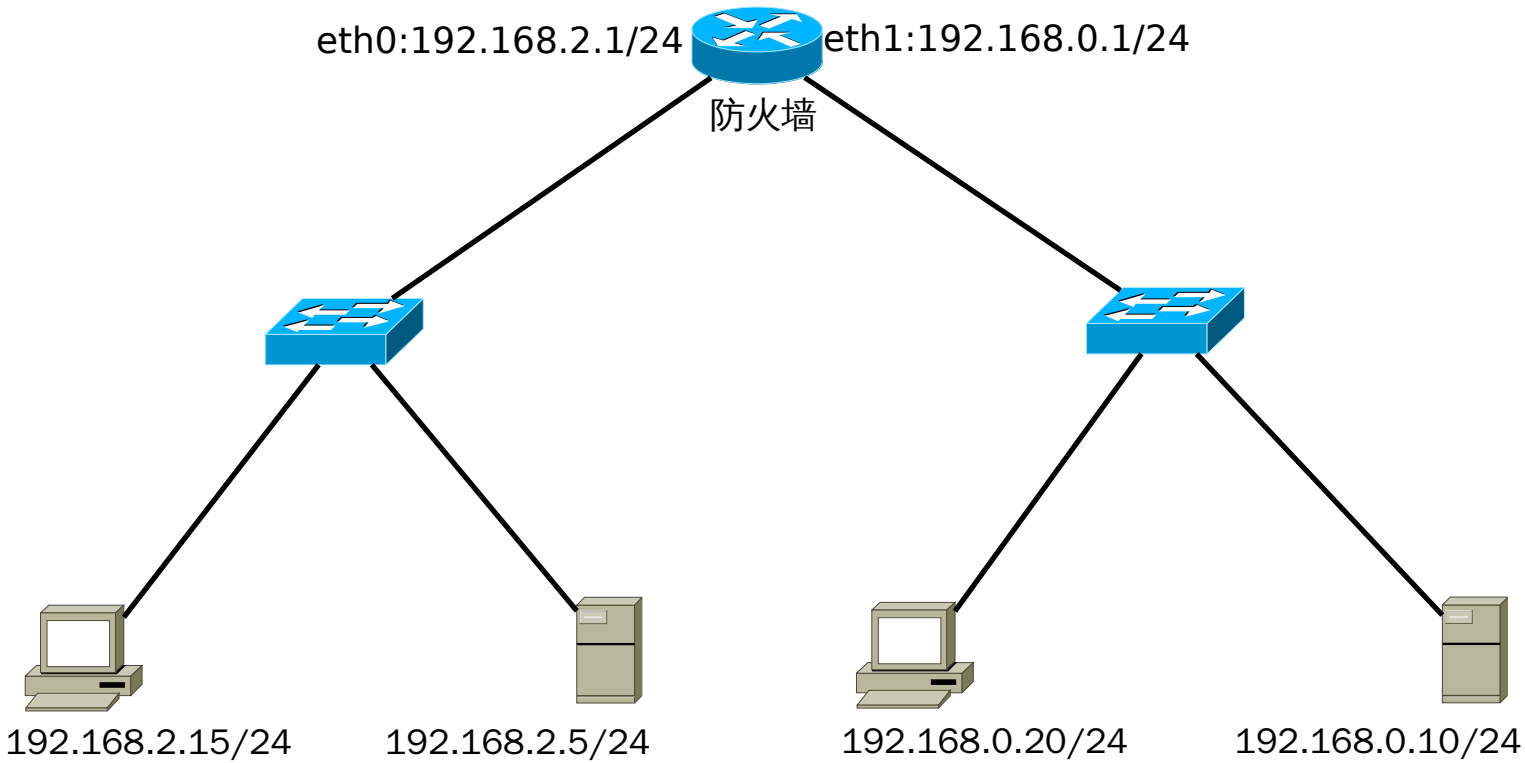
- 1. 如果在防火墙上执行 iptables -A INPUT -p icmp -j DROP，请问 192.168.2.15 及 192.168.0.20 谁可以 ping 到防火墙?

iptables 命令练习 (2)



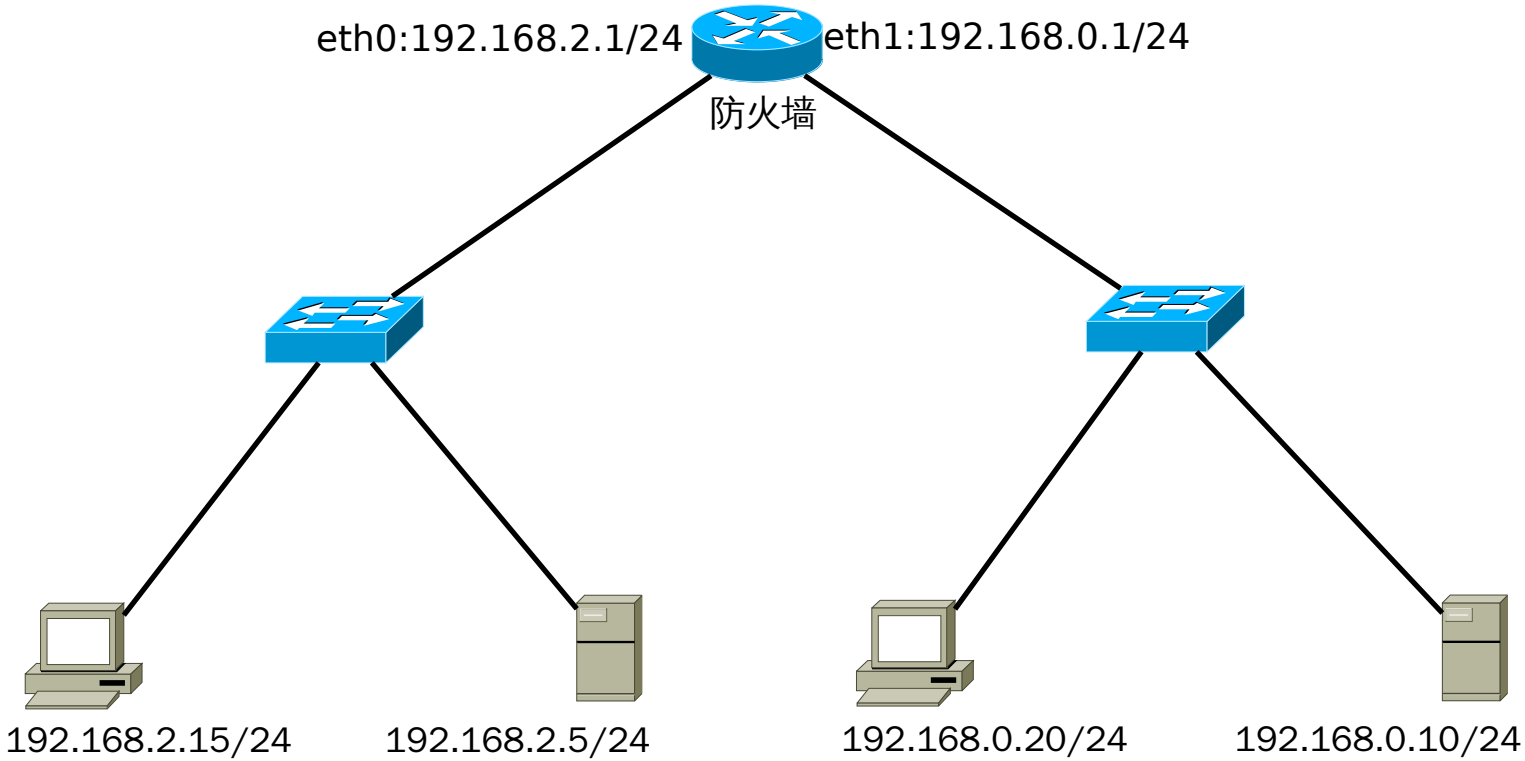
- 2. 如果在防火墙上执行 iptables -A INPUT -i eth0 -p icmp -d 192.168.0.2 -j DROP，请问 192.168.2.15 及 192.168.0.20 谁可以 ping 到防火墙?

iptables 命令练习 (3)



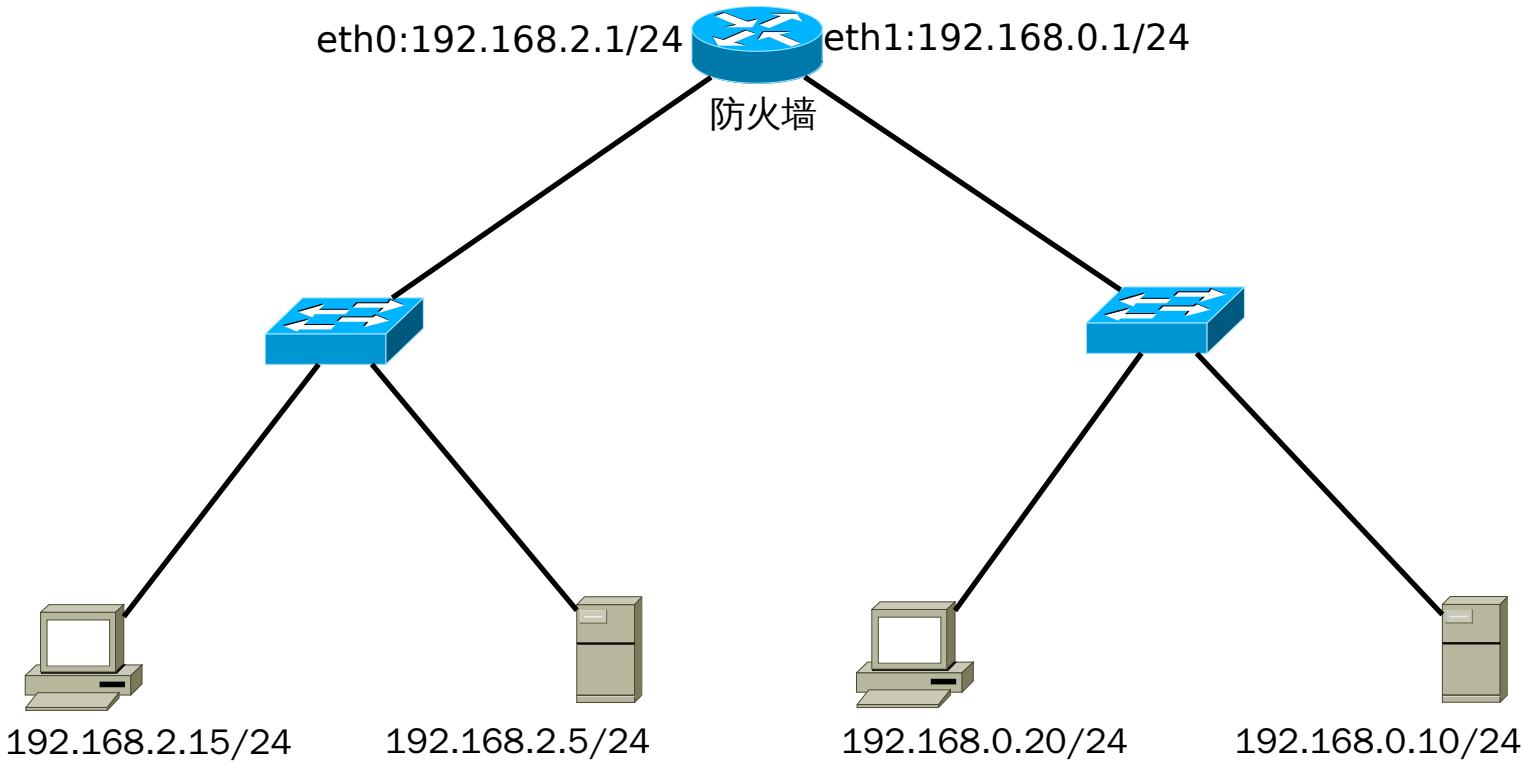
- 3. 如果在防火墙上执行 iptables -A INPUT -i eth1 –dport 80 -s 192.168.0.0/24 -j REJECT，假设防火墙上正在运行 Web 服务，请问哪些主机可以访问该 Web 服务?

iptables 命令练习 (4)



- 4. iptables -A INPUT -i eth1 -p tcp -d 192.168.2.5 –dport 80 -j REJECT，假设 192.168.2.5 是 Web 服务器，请问 192.168.2.15 及 192.168.0.20 哪一台主机可以访问该服务器?

iptables 命令练习 (5)



- 5. iptables -A FORWARD -i eth0 -o eth1 -p tcp –dport 80 -j REJECT，假设 192.168.2.5 和 192.168.0.10 都是 Web 服务器，请问:192.168.0.20 可以访问哪台 Web 服务器? 192.168.2.15 呢?

iptables 命令的基本使用 (1)

- 定义 iptables 规则的基本原则
  - 通常先拒绝所有数据包，再允许部分数据包，或反之；
  - 规则尽可能简单，能用一条解决的，就不要用多条；
  - 注意规则顺序：特殊规则放前面，通用规则放后面。
- 保存 iptables 规则
  - 用 iptables 命令创建的规则将自动保存到内存中，以 root 身份执行以下命令可永久保存规则：

    ```
    service iptables save # 保存至/etc/sysconfig/iptables
    ```

  - 也可以将 iptables 规则保存至指定文件：

    ```
    iptables-save 文件路径名
    ```

- 管理 iptables 服务

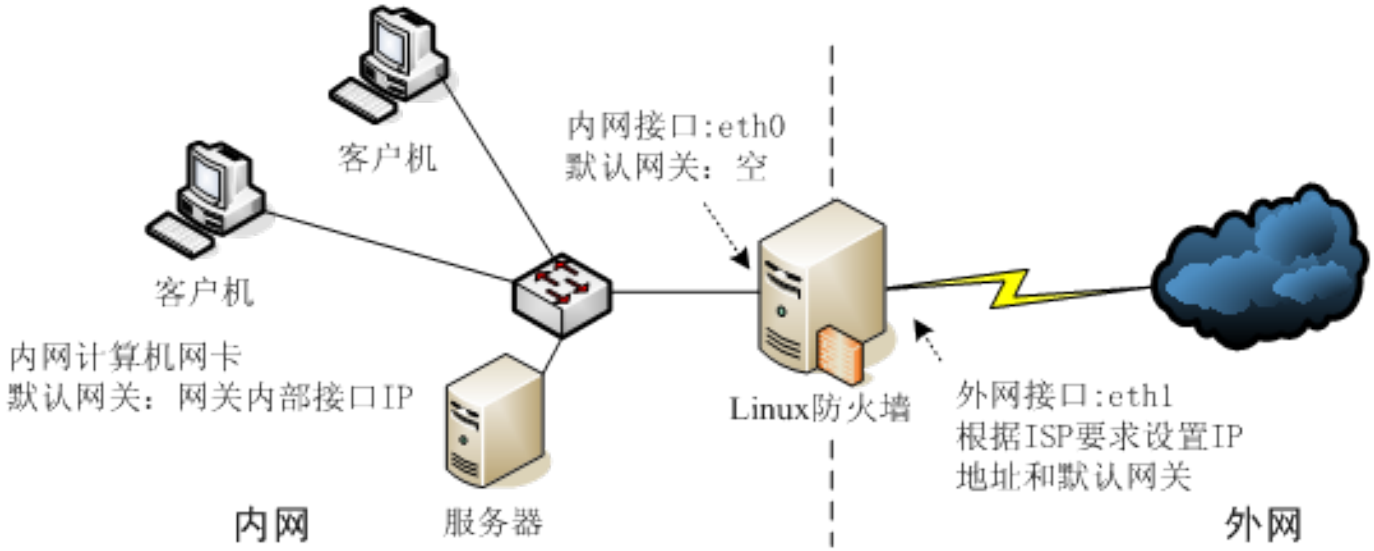
```
service iptables {start|stop|restart|status|panic|save}
#panic：丢弃所有防火墙规则，所有表中的策略都被设为 DROP
```

iptables 命令的基本使用 (2)

- iptables 控制脚本配置文件/etc/sysconfig/iptables-config
  - IPTABLES_MODULES: 指定一组空间独立的额外 iptables 模块在激活防火墙时加载。
  - IPTARLES_MODULES_UNLOAD: 在重新启动和停止时是否卸载模块。
  - IPTABLES_SAVE_ON_STOP: 停止防火墙时是否将当前的防火墙规则保存到/etc/sysconfig/iptables 文件。
  - IPTABLES_SAVE_ON_RESTART: 当防火墙重启时是否保存当前的防火墙规则。
  - IPTABLES_SAVE_COUNTER: 保存并恢复所有链和规则中的数据包和字节计数器。
  - IPTABLES_STATUS_NUMERIC: 输出的 IP 地址是数字格式还是域名 (主机名)。

iptables 防火墙基本配置 (1)

- 1. 配置网络环境



iptables 防火墙基本配置 (2)

- 2. 清除原有规则和计数器

```
iptables -F; iptables -X; iptables -Z
iptables -t nat -F
iptables -t nat -X
iptables -t nat -Z
```

- 3. 设置默认策略

```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P OUTPUT ACCEPT
iptables -t nat -P POSTROUTING ACCEPT
```

- 3. 保存规则并启用

```
service iptables save; service iptables restart
```

在防火墙上开放必要的内外网间通信

- 1. 允许回环地址通信

```
iptables -I INPUT 1 -i lo -j ACCEPT
iptables -I OUTPUT 1 -o lo -j ACCEPT
```

- 2. 开放防火墙上的端口

```
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT
```

- 3. 允许通过 SSH 管理防火墙

```
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT
# 以下命令仅允许从外网 (eth1) 访问防火墙
iptables -A INPUT -p tcp -i eth1 --dport 22 -j ACCEPT
iptables -A OUTPUT -p tcp -o eth1 --sport 22 -j ACCEPT
```

通过 NAT 方式共享上网 (1)

- 1. 服务器端 NAT 设置 (1)
  - 可通过定义 nat 表的 POSTROUTING 链来实现共享网络连接。一般为防火墙配置 IP 伪装 (MASQUERADE)，将发出请求的内网节点地址转换为防火墙设备 (例中为 eth1):

    ```
    iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
    ```

  - IP 伪装适合动态源地址转换，如果防火墙外网接口使用动态 IP 地址 (例如采用拨号方式或 DHCP 接入 Internet)，必须使用 MASQUERADE 方式：

    ```
    iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
    ```

通过 NAT 方式共享上网 (2)

- 1. 服务器端 NAT 设置 (2)
  - 源 NAT(SNAT) 和 IP 伪装都可以实现多台主机共享一个 Internet 连接，作用是一样的。如果防火墙外网接口使用静态 IP 地址，可以直接使用源 NAT 方式，定义如下:

```
iptables -t nat -A POSTROUTING -o eth1 \
-j SNAT --to 172.16.0.10
```

  - 还可以进一步限制共享连接的内网地址，例如：

```
iptables -t nat -A POSTROUTING -o eth1 \
-s 192.168.0.0/24 -j MASQUERADE
```

通过 NAT 方式共享上网 (3)

- 2. 调整防火墙包转发规则
  - 简单方法：将 FORWARD 链的默认策略该为允许

```
iptables -P FORWARD ACCEPT
```

  - 规范方法：保留 DROP 默认策略，添加相应允许规则

```
# 允许为整个内网 (eth0) 转发分组
iptables -A FORWARD -i eth0 -j ACCEPT
iptables -A FORWARD -o eth0 -j ACCEPT
```

  - 可根据需要设置仅允许转发特定协议的包，如 DNS 和 HTTP：

```
iptables -A FORWARD -p tcp --dport 53 -j ACCEPT
iptables -A FORWARD -p tcp --sport 53 -j ACCEPT
iptables -A FORWARD -p udp --dport 53 -j ACCEPT
iptables -A FORWARD -p udp --sport 53 -j ACCEPT
iptables -A FORWARD -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -p tcp --sport 80 -j ACCEPT
```

通过 NAT 方式共享上网 (4)

- 3. 客户端 NAT 设置
  - 只要设置 NAT 客户端的默认网关为 NAT 服务器 eth0 的 IP 地址,DNS 设为 ISP 的 DNS 服务器就可上网了。

通过端口映射发布内网服务器 (1)

- 1. 定义 NAT 端口映射
  - 可以使用 nat 表的 PREROUTING 链的-j DNAT 目标来定义转发传入数据包 (请求连接到内网服务) 的目标 IP 地址或端口。

```
iptables -t nat -A PREROUTING -i eth1 \
-p tcp --dport 80 -j DNAT --to 192.168.0.1:80
```

- 2. 调整包转发规则
  - 如果定义 FORWARD 链的“DROP”默认策略, 可使用以下规则 (防火墙内网接口为 eth0) 允许为内外网之间转发包。

```
iptables -A FORWARD -i eth0 -j ACCEPT
iptables -A FORWARD -o eth0 -j ACCEPT
```

  - 如果只转发传入 HTTP 请求, 可将规则修改为:

```
iptables -A FORWARD -i eth1 -p tcp --dport 80 \
-d 192.168.0.1 -j ACCEPT
iptables -A FORWARD -o eth1 -p tcp --sport 80 \
-s 192.168.0.1 -j ACCEPT
```

通过端口映射发布内网服务器 (2)

- 发布多台 Web 服务器
  - 可在防火墙上利用多端口发布多个 Web 服务器。例如：

```
iptables -t nat -A PREROUTING -i eth1 -p tcp \
--dport 80 -j DNAT --to 192.168.0.1:80
iptables -t nat -A PREROUTING -i eth1 -p tcp \
--dport 8000 -j DNAT --to 192.168.0.20:80
```

- 发布其他服务器
  - 以 FTP 服务器为例，使用以下规则：

```
iptables -t nat -A PREROUTING -i eth1 -p tcp \
--dport 20 -j DNAT --to 192.168.0.1:20
iptables -t nat -A PREROUTING -i eth1 -p tcp \
--dport 21 -j DNAT --to 192.168.0.1:21
```

防止恶意软件和假冒 IP 地址

- 可以限制访问服务器带来的恶意应用程序，如木马、蠕虫和其他客户/服务器病毒。例如，一些木马在端口 31337~31340(黑客术语称为“elite”端口) 扫描网络服务。以下规则丢弃试图使用 31337 端口的所有 TCP 数据包:

```
iptables -A OUTPUT -o eth1 -p tcp \
--dport 31337 --sport 31337 -j DROP
iptables -A FORWARD -o eth1 -p tcp \
--dport 31337 --sport 31337 -j DROP
```

- 也可阻断试图仿冒内网 IP 地址攻击内网的外部连接。例如，如果内网用户使用 192.168.0.0/24 网段，可以设计一条规则指示外网接口 (如 eth1) 丢弃到达该接口的数据包 (由内网 IP 段的私有 IP 发出)

```
iptables -A FORWARD -s 192.168.0.0/24 -i eth1 -j DROP
```

- 如果将拒绝转发数据包作为默认策略，任何到外部设备的假冒 IP 地址自动被拒绝。

配置状态防火墙

- 连接跟踪可以让 Netfilter 获知某个特定连接的状态。
- iptables 可设置以下 4 种连接状态：
  1. NEW: 表示匹配的数据包正在创建一个新连接
  2. ESTABLISHED: 表示匹配的数据包属于某个已经建立的双向传送的连接
  3. RELATED: 表示匹配的数据包正在启动一个与现有连接相关的新连接
  4. INVALID: 表示匹配的数据包不能与一个已知的连接相关联，通常应丢掉
- 状态匹配由 state 模块提供，使用时需要使用选项-m 加载，以下例子表示，通过连接跟踪仅转发同已建立连接相关的数据包，如 FTP-DATA 数据连接:

```
iptables -A FORWARD -m state \
--state ESTABLISHED,RELATED -j ACCEPT
```

配置非军事区 (DMZ)

- DMZ 是一个非安全系统与安全系统之间的缓冲区，缓冲区位于内外网之间的特殊子网，可部署一些要公开的服务器。
- 需创建 iptables 规则，将数据包路由到位于 DMZ 的服务器。
- 例如，将 HTTP 请求路由到 HTTP 服务器 10.0.0.2(位于内网 192.168.1.0/24 的外面):

```
iptables -t nat -A PREROUTING -i eth1 -p tcp \
--dport 80 -j DNAT --to-destination 10.0.0.2:80
```

- 如果 HTTP 服务器配置为接收 SSL 安全连接，端口 443 也必须转发：

```
iptables -t nat -A PREROUTING -i eth1 -p tcp \
--dport 443 -j DNAT --to-destination 10.0.0.2:80
```

安装和管理 squid 服务

- 安装 squid 软件包

```
yun install squid
```

- 管理 squid 服务

```
service squid {start|stop|status|reload|restart|\
condrestart}
```

squid 配置文件/etc/squid/squid.conf(1)

- 认证选项
  - Squid 支持多种用户认证模式, 如基本、摘要 (Digest)、NTLM 和协商 (Negotiate), 指定如何从客户端接受用户名和密码。指令 auth 用于定义不同认证模式的参数:

    auth_param 模式 参数 [设置]
- 访问控制选项
  - Squid 默认拒绝所有访问客户端的请求，为了能让客户端通过代理服务器访问，最简单的方法就是定义一个针对客户端 IP 地址的访问控制列表 (ACL)，并允许来自这些地址的 HTTP 请求。

    acl 访问控制列表名称 访问控制列表类型 字符串 1 ...
    http_access allow|deny [!]访问控制列表名称 ...
- 网络选项
  - 网络选项 http 指定 Squid 监听客户端 HTTP 请求的 IP 地址和端口, 语法格式为:

    http_port [主机名或 IP:] 端口 [选项]

squid 配置文件/etc/squid/squid.conf(2)

- **相邻缓存服务器选项：用于设置多层缓存**

```
# 指定多层缓存网络中其他缓存服务器
cache_peer hostname type http_port icp_port [options]
cache_peer p.abc.com parent 3128 3130 proxy-only defaul
cache_peer s1.abc.com sibling 3128 3130 proxy-only
cache_peer s2.abc.com sibling 3128 3130 proxy-only
#type: parent(父级)、sibling(同级)
#http_port: 该缓存服务器监听客户端 http 请求的端口，默认
#icp_port：该缓存服务器 ICP 查询所用端口，默认 3130
#options：proxy-only: 不保存来自缓存的对象
#          default：作为顶层缓存服务器
# 限定要查询的邻居缓存服务器的域
cache_peer_domain p.abc.com [!].edu
# 通过 acl 提供更灵活的访问控制
cache_peer_access p.abc.com allow|deny acl1
```

squid 配置文件/etc/squid/squid.conf(3)

- 内存缓存选项

```
cache_mem 8 MB #squid 可以使用的内存大小
# 内存缓存中可保存的最大对象
maximum_object_size_in_memory 8KB
```

- 硬盘缓存选项

```
# 指定缓存空间的类型，位置，大小及其目录结构
cache_dir ufs /var/spool/squid 100 16 256
cache_swap_low 90   # 交换空间上限
cache_swap_high 95 # 交换空间下限
maximum_object_size 4096KB # 硬盘缓存中可保存的最大对象
```

squid 配置文件/etc/squid/squid.conf(4)

- 日志文件路径
  - logformat：定义访问日志文件格式
  - access_log：设置记录客户请求的日志文件
  - cache_log：设置 squid 产生的一般信息的日志文件
  - cache_store_log：设置记录对象存储情况的日志文件
- 管理参数
  - cache_mgr：设置管理员的邮件地址
  - cache_effective_user 与 cache_effective_group：设置 squid 启动后，以何用户身份运行，默认设置为 squid
  - visible_hostname：定义在返回给用户的出错信息中所显示的主机名

squid 配置文件/etc/squid/squid.conf(5)

- 超时设置
  - connect_timeout：设置 Squid 等待连接完成的超时值，默认为 1 分钟
  - peer_connect_timeout：设置连接其他缓存服务器的超时值，默认为 30 秒
  - read_timeout：如果客户在指定时间内，未从 Squid 服务器读取任何数据，则 Squid 将终止该客户的请求，默认为 15 分钟
  - request_timeout：设置 Squid 与客户端建立连接后，等待客户发出 HTTP 请求的超时时间，默认设置为 5 分种
  - persistent_request_timeout：设置在前一个连接请求完成后，在同一个连接上等待下一个新的 HTTP 请求的超时时间，默认设置为 1 分钟
  - ident_timeout：设置 Squid 等待用户认证请求的超时时间，默认为 10 秒

squid 命令行

- squid 命令可用于管理和调试，主要选项如下
  - -f file：用指定配置文件取代默认配置文件
  - -k：让 squid 执行指定管理功能

```
reconfigure  # 重新读取配置文件
shutdown     # 关闭
debug        # 进入调试模式
check        # 检查 squid 进程
parse        # 检查配置文件
```

  - -u port：指定 ICP 端口号，取代配置文件内的 icp_port
  - -z：初始化缓存，首次运行 squid 或者增加新的缓存目录时，必须使用该选项！

配置标准代理服务器 (1)

```
vim /etc/squid/squid.conf
http_port 192.168.0.2:3128 # 监听地址及端口
cache_mem 128 MB # 设置高速缓存为 128MB
cache_dir ufs /var/spool/squid 4096 16 256 # 设置硬盘缓存
access_log /var/log/squid/access.log # 设置访问日志
cache_log /var/log/squid/cache.log # 设置缓存日志
cache_store_log /var/log/squid/store.log # 设置网页缓存日志
dns_nameservers 211.137.191.26 # 设置 DNS 服务器
acl all src 0.0.0.0/0.0.0.0 # 定义访问控制列表 all
http_access allow all # 允许所有客户端访问
cache_mgr root@abc.com # 设置管理员 E-mail 地址
cache_effective_user squid # 设置 squid 进程所有者
cache_effective_group squid # 设置 squid 进程所属组
visible_hostname 192.168.0.2 # 设置 squid 可见主机名
```

配置标准代理服务器 (2)

- 如果 Linux 服务器开启了防火墙功能，还需要关闭防火墙功能，或者允许访问代理服务器端口 (例中为 3128。

```
iptables -A INPUT -p tcp --dport 3128 -j ACCEPT
```

- 配置代理客户端
  - 打开 Firefox 浏览器中，选择“编辑”>“首选项”菜单打开相应的对话框，在“常规”选项卡中单击“连接设置”按钮，然后选中“手动配置代理”单选钮，在“HTTP 代理”和“端口”文本框中输入要使用的代理服务器的 IP 地址及端口号。
  - Windows 平台上一般使用 IE 浏览器。打开 IE 浏览器，选择“工具”>“Internet 选项”菜单打开相应的对话框，切换到“连接”选项卡，单击“局域网设置”按钮，然后选中“为 LAN 使用代理服务器”复选框，输入代理服务器正确的 IP 地址及端口号。

squid 服务器访问控制 (1)

- 允许某内部网段用户访问

```
acl mynet src 192.168.0.0/255.255.0.0
http_access allow mynet
http_access deny all
```

- 禁止某内部网段用户访问

```
acl othernet src 192.168.10.0/255.255.0.0
http_access deny othernet
```

- 站点屏蔽
  屏蔽某些特定站点或含有某些特定字词的站点，例如：

```
acl sexurl url-regex -i sex
http_access deny sexurl
```

squid 服务器访问控制 (2)

- 下载内容屏蔽
  禁止客户机下载 *.mp3、*.exe、*.zip 和 *.rar 类型的文件。

```
acl bigfiles urlpath_regex -i \.mp3$ \.exe$ \zip$ \.rar
http_access deny bigfiles
```

- 限制访问时段
  允许所有用户在周一至周五 8:30~20:30 访问代理服务器，允许管理员每天下午访问代理服务器，其他时段一律拒绝访问。

```
act all src 0.0.0.0/0.0.0.0
acl administrator 192.168.0.0/24
acl common_time time MTWHF 8:30-20:30
acl manage_time time MTWHFAS 13:00-18:00
http_access allow all common_time
http_access allow administrator manage_time
http_access deny all
```

squid 服务器访问控制 (3)

- 设置 CONNECT
  有些用户通过二级代理软件访问非法站点，可在 Squid 中通过 CONNECT 方法来拒绝访问。

```
acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl to_localhost dst 127.0.0.0/8
acl SSL_ports port 443
acl Safe_ports port 80 # http
acl Safe_ports port 21 # ftp
acl Safe_ports port 443 # https
acl Safe_ports port 1025-65535 # unregistered ports
acl Safe_ports port 777 # multiling http
acl CONNECT method CONNECT
```

squid 服务器访问控制 (4)

- 设置 CONNECT(续)

```
http_access allow manager localhost
http_access deny manager
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
```

squid 服务器用户认证 (1)

- 在 Squid 配置文件中设置认证选项
  - (1) 设置要使用的认证程序及其相关选项，在配置文件开头部分设置以下选项。

```
# 定义认证方式、认证程序路径和需读取的账户文件
auth_param basic program \
/usr/lib/squid/ncsa_auth /etc/squid/passwd
# 设置认证程序的进程数
auth_param basic children 5
# 设置认证有效时间，超过该时间要求重新输入用户名和密码
auth_param basic credentialsttl 2 hours
# 设置认证领域内容，即浏览器显示认证对话框时的提示内容
auth_param basic realm \
This is a Squid proxy-caching
```

squid 服务器用户认证 (2)

- 在 Squid 配置文件中设置认证选项 (续)
  - (2) 为认证用户设置 ACL，并给出访问控制规则，在配置文件中设置以下选项。

```
acl noauth_user src 192.168.0.21
acl auth_user proxy_auth REQUIRED
http_access allow noauth_user
http_access allow auth_user
```

squid 服务器用户认证 (3)

- 建立账户文件
  - 为建立供用户认证使用的账户文件，可利用 Apache 的 htpasswd 程序生成账户文件/etc/squid/passwd，该账户文件每行包含一个用户的信息，即用户名和经过加密后的密码。

```
htpasswd -c /etc/squid/passwd mike # 添加首个用户
htpasswd /etc/squid/passwd mary # 添加后续用户

service squid restart
```

  - 测试用户认证
    - 重启 squid 服务
    - 配置客户端
    - 测试访问网页
  - 注意：squid 不支持在透明代理模式下启用用户身份认证功能，但在正、反向代理模式下都有效。

配置透明代理服务器 (1)

- 概述
  - 所谓透明，是指客户端感觉不到代理的存在，不需要对浏览器进行代理设置。
  - 在 squid 中，透明代理又称 Interception Caching(拦截缓存)，或缓存重定向。
  - 透明代理服务器阻断网络通信，并且过滤出访问外网的 HTTP(80 端口) 流量。如果客户端的请求在本地有缓存，则将缓存的数据直接发给用户。
  - 对于 Linux 服务器来说，将 iptables 与 Squid 服务器结合起来可实现这种透明代理。Squid 支持 HTTP 和 FTP 等协议的代理，在缓存数据的同时，也缓存 DNS 查询结果，支持 SSL 和访问控制；对于 Squid 无法代理的服务，则可通过 iptables 共享连接来实现，同时具有缓存功能，加速 Web 的访问。

配置透明代理服务器 (2)

- 透明代理配置过程

```
#1. 修改 squid.conf, 启用透明代理支持
http_port 192.168.0.2:3128 transparent
#2. 启用 ip 转发功能
#3. 配置 iptables, 实现 http 流量拦截
iptables -t nat -A PREROUTING -i eth0 -p tcp \
--dport 80 -j REDIRECT --to-ports 3128
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
#4. 重启 squid 或重新加载 squid 配置文件
#5. 重启 iptables
#6. 客户端只需要将默认网关设置为 squid 主机的内网地址,
#   即可正常访问外部网站
```

配置反向代理服务器 (1)

- 配置基本的反向代理服务器
  - 关于反向代理的选项必须出现在 squid.conf 配置文件开头，在其他转发代理配置 (如 http 等) 之前，否则标准代理访问规则将阻止访问加速站点 (反向代理服务器)。

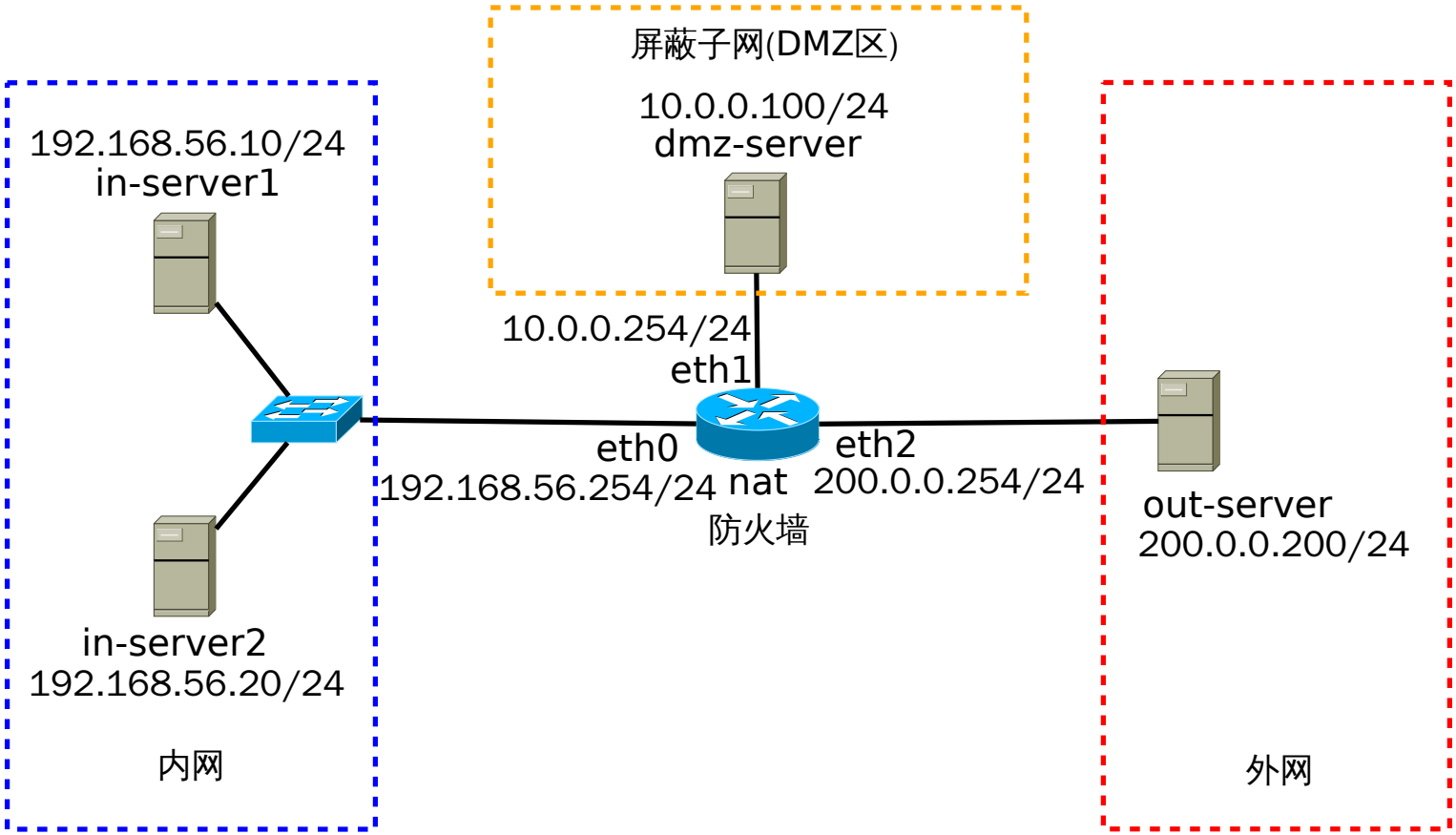
```
# 设置反向代理监听端口 (通常为 80) 和反向代理模式
http_port 80 accel
# 设置所代理的内部 web 服务器
cache_peer 192.168.0.21 parent 80 0 \
no-query originserver name=myAccel
# 设置允许访问该加速站点，不允许其他访问
acl our_sites dstdomain www.abc.name
http_access allow our_sites
cache_peer_access myAccel allow our_sites
cache_peer_access myAccel deny all
# 重启 squid 服务
# 将 web 网站的公共域名执行 squid 服务器后进行访问测试
```

配置反向代理服务器 (2)

- 多台服务器配置
  - 例：两台内网服务器 (192.168.0.2 和 192.168.0.21),squid 的内网接口 eth0(192.168.0.2), 外网接口 eth1(172.16.0.10)。

```
http_port 80 vhost # 设置监听端口，并启用反向代理
# 指定 Squid 要代理的内部 Web 服务器
cache_peer 192.168.0.2 parent 80 0 \
no-query originserver name=a
cache_peer 192.168.0.21 parent 80 0 \
no-query originserver name=b
cache_peer_domain a www.servera.com
cache_peer_domain b www.serverb.com
# 设置访问控制，允许所有外部客户端访问这些站点
acl all src 0.0.0.0/0.0.0.0
http_access allow all
cache_peer_access a allow all
cache_peer_access b allow all
# 重启 squid，将公共域名指向 Squid 代理服务器并测试
```

案例网络拓扑结构图


屏蔽子网(DMZ区)
10.0.0.100/24
dmz-server
192.168.56.10/24
in-server1
10.0.0.254/24
eth1
eth0
192.168.56.254/24
nat
防火墙
eth2
200.0.0.254/24
out-server
200.0.0.200/24
in-server2
192.168.56.20/24
内网
外网

案例需求

- 内网、外网
  - 搭建 web 服务器
  - 配置 dns 客户端
- dmz 区
  - 为内外网客户提供 dns 服务 (利用视图)
    - 内网：abc.com
    - 外网：xyz.net
  - 提供 ftp 服务
  - 为内网客户提供 http 透明代理上网服务
  - 为内网服务器提供 http 反向代理服务
- 防火墙
  - 提供数据包过滤服务
  - 提供内网访问外网服务器的 SNAT 功能
  - 提供外网访问内网服务器的 DNAT 功能
  - 提供内网透明代理的 http 劫持功能

案例实施 (1)

- 1. 配置主机名、IP 地址等，确保网络通畅
  - 注意，为了让内网主机和 dmz 区主机分别通过 NAT 转换为外部 IP 地址，需要为防火墙的外网接口再增加一个 IP 地址，可配置 eth2:0 子接口 ip 地址为 200.0.0.253

```
cd /etc/sysconfig/network-scripts
vim ifcfg-eth2:0
DEVICE=eth2:0
BOOTPROTO=none
IPADDR=200.0.0.253
NETMASK=255.255.255.0
ONBOOT=yes
```

  - 内网源地址通过 NAT 转换为 200.0.0.254
  - DMZ 区源地址通过 NAT 转换为 200.0.0.253

案例实施 (2)

- 2. 在 dmz-server 服务器上配置 DNS 服务 (1)

```
vim /var/named/chroot/etc/named.conf
options {
        listen-on port 53 { any; };
        directory       "/var/named";
        allow-query     { any; };
        allow-query-cache { any; };
};
# 定义内网用户访问控制列表
acl "inside_hosts" {
        127.0.0.1;192.168.56.0/24;10.0.0.0/24;
};
```

案例实施 (3)

- 3. 在 dmz-server 服务器上配置 DNS 服务 (2)

```
# 定义内网用户视图
view inside {
        match-clients       { inside_hosts };
        match-destinations { any; };
        recursion yes;
        include "/etc/named.rfc1912.zones";
        zone "abc.com" IN {
                type master;
                file "abc.com.zone.in";
        };
        zone "xyz.net" IN {
                type master;
                file "xyz.net.zone";
        };
};
```

案例实施 (4)

- 3. 在 dmz-server 服务器上配置 DNS 服务 (3)

```
# 定义其他用户的视图
view outside {
        match-clients  { any; };
        match-destinations { any; };
        recursion no;
        include "/etc/named.rfc1912.zones";
        zone "abc.com" IN {
                type master;
                file "abc.com.zone.out";
        };
        zone "xyz.net" IN {
                type master;
                file "xyz.net.zone";
        };
};
```

案例实施 (5)

- 3. 在 dmz-server 服务器上配置 DNS 服务 (4)

```
cd /var/named/chroot/var/named
vim abc.com.zone.in
$TTL 86400
@       IN SOA ns1 root (
                                42      ; serial (d. adams)
                                3H      ; refresh
                                15M     ; retry
                                1W      ; expiry
                                1D )    ; minimum
        IN NS                   ns1
ns1     IN A                    10.0.0.100
www     IN A                    192.168.56.10
        IN A                    192.168.56.20
ftp     IN CNAME                www
```

案例实施 (6)

- 3. 在 dmz-server 服务器上配置 DNS 服务 (5)

```
vim abc.com.zone.out
$TTL 86400
@       IN SOA ns1 root (
                                42      ; serial (d. adams)
                                3H      ; refresh
                                15M     ; retry
                                1W      ; expiry
                                1D )    ; minimum
        IN NS                   ns1
ns1     IN A                    200.0.0.253
www     IN A                    200.0.0.253
ftp     IN CNAME                www
```

案例实施 (7)

- 3. 在 dmz-server 服务器上配置 DNS 服务 (6)

```
vim xyz.net.zone
$TTL 86400
@       IN SOA ns1 root (
                          42        ; serial (d. adams)
                          3H        ; refresh
                          15M       ; retry
                          1W        ; expiry
                          1D )      ; minimum
        IN NS             ns1
ns1     IN A              10.0.0.100
www     IN A              200.0.0.200
ftp     IN CNAME          www

chkconfig --level 345 named on; service named start
```

案例实施 (8)

- 4. 配置 dns 客户端
  - 内网和 DMZ 区客户端

```
vim /etc/resolv.conf
nameserver 10.0.0.100
search localdomain
```

  - 外网客户端

```
vim /etc/resolv.conf
nameserver 200.0.0.253
search localdomain
```

  - 测试：内网和 DMZ 区客户端可以解析域名，外网客户端暂时无法解析域名。

案例实施 (9)

- 5. 配置防火墙，使外网客户端可以访问 DMZ 区服务器

```
vim firewall.sh
iptables -F; iptables -Z
iptables -t nat -F; iptables -t nat -Z
# 对 http 服务的访问转向 DMZ 区反向代理
iptables -t nat -A PREROUTING -i eth2 -p tcp \
--dport 80 -j DNAT --to 10.0.0.100:80
# 对 ftp 服务的访问转向 DMZ 区 ftp 服务器
iptables -t nat -A PREROUTING -i eth2 -p tcp \
--dport 21 -j DNAT --to 10.0.0.100:21
# 对 dns 服务的访问转向 DMZ 区 dns 服务器
iptables -t nat -A PREROUTING -i eth2 -p udp \
--dport 53 -j DNAT --to 10.0.0.100:53
service iptables save
```

案例实施 (10)

- 6. 阶段测试
  - 外网客户端再次测试 dns 服务
  - 外网服务器安装启动 http 服务
  - dmz 区服务器安装启动 ftp 服务
  - 内网服务器安装启动 http 服务 (设置基于 IP 的虚拟主机)
  - 测试内、外网访问 ftp 服务
  - 测试内访问外网 http 服务
  - 测试外网访问内网 http 服务

案例实施 (11)

- 7. 配置 DMZ 区服务器对内网 http 服务的反向代理 (1)
  - 配置 squid 服务器

```
cd /etc/squid/
cp -p squid.conf squid.conf.bak
sed -i '/^#\|^$/d' squid.conf # 去除注释行和空行
vim squid.conf  # 在最前面添加 (反向代理必须设在最前面)
visible_hostname 10.0.0.100
http_port 80 vhost
cache_peer 192.168.56.10 parent 80 0 \
no-query originserver name=a
cache_peer_domain a www.abc.com
acl all src 0.0.0.0/0.0.0.0
http_access allow all
cache_peer_access a allow all
```

案例实施 (12)

- 7. 配置 DMZ 区服务器对内网 http 服务的反向代理 (2)
  - 启动 squid 服务器

```
squid -k parse # 检查配置文件是否正确
squid -z       # 初始化 squid 缓存
service squid start # 启动 squid
```

  - 外网再次测试访问内外 http 服务器

案例实施 (13)

- 8. 配置 DMZ 区服务器为内网客户端的 http 透明代理 (1)
  - 配置防火墙将内网 http 数据包重定向至透明代理

```
iptables -t nat -A PREROUTING -i eth0 -p tcp \
--dport 80 -j DNAT --to 10.0.0.100:3128
service iptables save
```

  - 测试内网访问外网

案例实施 (14)

- 8. 配置 DMZ 区服务器的 http 透明代理 (2)
  - 配置 squid 服务器

```
vim /etc/squid/squid.conf # 在反向代理的配置后面添加
http_port 10.0.0.100:3128 transparent # 透明代理
dns_nameservers 127.0.0.1 # 指定代理的 dns 服务器
acl inside src 192.168.56.0/255.255.255.0 # 定义内网列表
http_access allow inside # 允许内网访问透明代理
```

  - 重启 squid 服务器

```
service squid restart
```

  - 再次测试内网访问外网

案例实施 (15)

- 9. 通过防火墙对各种数据包进行限制
  - 可以根据具体的安全需求进一步加强防火墙的设置