第 8 讲 Linux 网络与服务配置管理

王晓庆

wangxiaoqing@outlook.com

May 20, 2016

Outline

网卡设备命名规则

- 网卡设备名的格式为：网卡类型 + 网卡序号
- 以太网卡的设备名用 ethN 来表示，第一块以太网卡的设备名为 eth0, 第二块以太网卡的设备名为 eth1，其余依次类推
- Linux 支持一块物理网卡绑定多个 IP 地址，此时对于每个绑定的 IP 地址，需要一个虚拟网卡，该网卡的设备名为 ethN:M，其中 N 和 M 均为从 0 开始的数字

网络配置文件

- /etc/hosts

  存储主机名和 IP 地址映射，用来解析无法用其他方法解析的主机名
- /etc/resolv.conf

  与域名解析有关的设置
- /etc/sysconfig/network

  定义所有网络接口的路由和主机信息
- /etc/sysconfig/network-scripts/ifcfg-< 网络接口名 >

  对于每个网络接口，都有一个相应的接口配置文件，提供该网络接口的特定信息

网卡基本配置

- 编辑配置文件

```
vim /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0                    # 该网卡设备名称
ONBOOT=yes                     # 计算机启动时是否启用该网卡
BOOTPROTO=static               # 启动协议 (dhcp/static/none)
IPADDR=192.168.56.200          # IP 地址
NETMASK=255.255.255.0          # 子网掩码
HWADDR=00:0c:29:ec:a8:50       # 物理地址 (MAC)
GATEWAY=192.168.0.1            # 默认网关地址
TYPE=Ethernet                  # 网卡类型
```

- 使用 setup 命令

```
setup
```

- 上述配置将永久生效，但需要重启网络服务才能生效

```
service network restart
```

常用网络配置命令

- ifconfig

```
ifconfig -a                    # 查看所有网络接口信息
ifconfig eth0                  # 查看指定网络接口信息
# 配置接口地址，立即生效，但未修改配置文件
ifconfig eth0 192.168.56.201 netmask 255.255.255.0
ifconfig eth0 down|up   # 激活/关闭网卡
```

- ifup/ifdown

```
ifup|ifdown eth0          # 激活/关闭网卡
```

- route

```
route -n     # 查看路由表 (-n 不进行域名解析)
netstat -rn # 查看路由表
```

配置主机名和 DNS 客户端

- 配置主机名

```
hostname                # 显示主机名
hostname myserver       # 设置主机名 (立即生效，但非永久)
vim /etc/sysconfig/network # 永久设置主机名 (重启后生效)
```

- 配置 DNS 客户端

```
vim /etc/resolv.conf
nameserver 192.168.0.1 # 名称服务器
nameserver 8.8.8.8
search abc.com          #DNS 搜索路径
domain abc.com          # 默认域名
```

常用网络测试与监控命令

- ping、traceroute、netstat
- sar -n DEV|EDEV|NFS|NFSD|SOCK|ALL
- tcpdump

  tcpdump [选项] [-c 数量] [-F 文件名] [-i 网络接口] [-r 文件名] [-s snaplen] [-T 类型] [w 文件名] [表达式]

  - tcp 使用表达式过滤需要捕捉的数据包
    - 类型：host, net, port, portrange
    - 方向：src, dst, dst or src, dst and src(默认)
    - 协议：fddi, ether, ip, arp, rarp, tcp, udp
    - 其他：gateway, broadcast, less, greater
    - 逻辑运算符：and, or, not, &&, ||, !
  - 选项
    - -a 将网络地址和广播地址转换为名称
    - -e 输出数据链路层头部信息
    - -n 不进行地址解析
    - -v 显示详细信息

常用网络测试与监控命令

- tcpdump

```
tcpdump not (port 22 or port 23) # 不捕获 ssh 和 telnet
tcpdump -n -i eth0 not port 22 # 捕获 eth0 的非 ssh 包
tcpdump -i eth0 [src|dst] host 192.168.1.1 # 过滤主机
tcpdump -i eth0 [src|dst] net 192.168 # 过滤网络
tcpdump -i eth0 [src|dst] port 80 # 过滤端口号
tcpdump -i eth0 -c 8 icmp # 过滤协议，捕获 8 个包
tcpdump -i eth1 '((tcp) and (port 80) and
((dst host 192.168.1.254) or (dst host 192.168.1.200)))'
tcpdump -i eth1 '((icmp) and ((ether dst host
00:01:02:03:04:05)))'
tcpdump -i eth1 '((tcp) and ((dst net 192.168) and
(not dst host 192.168.1.200)))'
man tcpdump # 查看详细信息和更多例子
```

Linux 的路由功能

- Linux 作为网络操作系统，内核本身就提供数据包转发，支持基本的路由功能，搭配路由软件，则可以成为较为专业的路由器
  - Linux 主机用作软件路由器，至少需要安装两个网络接口
- 启用 Linux 内核路由转发功能
  - 默认情况下内核未开启 IP 数据包转发功能

```
sysctl -w net.ipv4.ip_forward=1 # 临时启用 IP 转发功能
vim /etc/sysctl.conf
net.ipv4.ip_forward=1           # 永久开启 IP 转发功能
sysctl -p                       # 使配置文件立即生效
```

配置静态路由 (1)

- 使用 route 命令配置静态路由

```
route add default gw 192.168.56.1 dev eth0 # 添加默认路
route del default gw 192.168.56.1 dev eth0 # 删除默认路
# 添加静态路由
route add -net 192.56.76.0/24 gw 192.168.56.200
route add -net 192.56.76.0/24 dev eth0
```

- 使用 ip route 命令配置静态路由

```
ip route add 192.0.2.0/24 via 10.0.0.1 [dev eth0]
```

配置静态路由 (2)

- 配置永久路由：
  /etc/sysconfig/network-scripts/route-ethn(n=0,1,2,...)
  - 配置格式 1

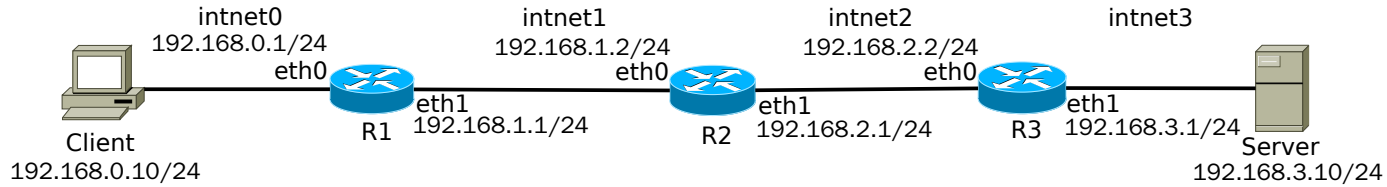
```
default 192.168.0.1 dev eth0
10.10.10.0/24 via 192.168.0.1 dev eth0
172.16.1.0/24 via 192.168.0.1 dev eth0
```

  - 配置格式 2

```
ADDRESS0=10.10.10.0
NETMASK0=255.255.255.0
GATEWAY0=192.168.0.1
ADDRESS1=172.16.1.0
NETMASK1=255.255.255.0
GATEWAY1=192.168.0.1
```

配置静态路由 (3)

- 配置实例



- 配置步骤
  - 配置主机和路由器的接口 IP 地址

```
ifconfig eth0 192.168.0.10
```

  - 配置主机的默认路由

```
route add default gw 192.168.0.1
```

  - 启用路由器的 ip 转发功能

```
sysctl -w net.ipv4.ip_forward=1
```

  - 配置路由器的静态路由

```
route add -net 192.168.3.0/24 gw 192.168.1.2
```

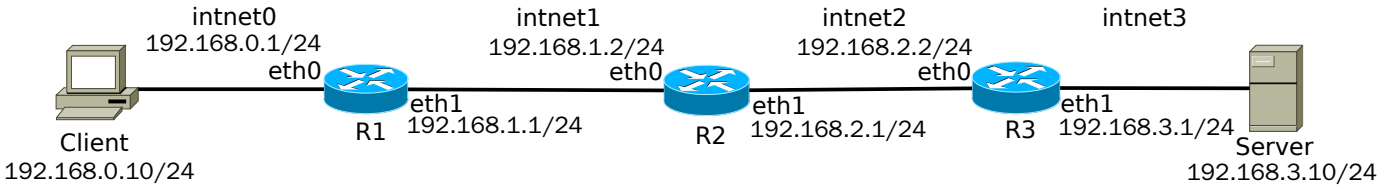
  - ping 测试连通性

配置动态路由 (1)

- zebra 是一个开源的 TCP/IP 路由软件，支持主流的动态路由协议 RIPvl、RIPv2、RIPng、OSPFv3、BGP-4 和 BGP-4+ 等
- zebra 作为软件路由器，其配置与 Cisco IOS 极其类似
- 在实际运行中必须先启动 zebra，再启动需要的动态路由协议
- 在 CentOS 5 上安装 quagga 之后，/etc/init.d 目录中将提供 zebra 相关的守护进程

配置动态路由 (2)

- 配置实例



- 配置步骤
  - 安装 quagga 软件包
  - 安装 telnet-server 软件包
  - 设置 zebra

```
vim /etc/quagga/zebra.conf # 修改 zebra 主配置文件
hostname R1
password abc
enable password 123
!log file zebra.log
service zebra start          # 启动 zebra 服务
telnet 127.0.0.1 2601        # 本机登录 zebra 服务
```

配置动态路由 (3)

- 配置步骤 (2)
  - 配置 RIP 协议

```
vim /etc/quagga/ripd.conf   # 配置 rip 服务
hostname R1
password abc
enable password 123
log stdout
router rip
  version 2
  network 192.168.0.0/24
  network 192.168.1.0/24
interface eth1 # 每个 rip 协议接口均需配置验证
  ip rip authentication mode md5
  ip rip authentication string abc
```

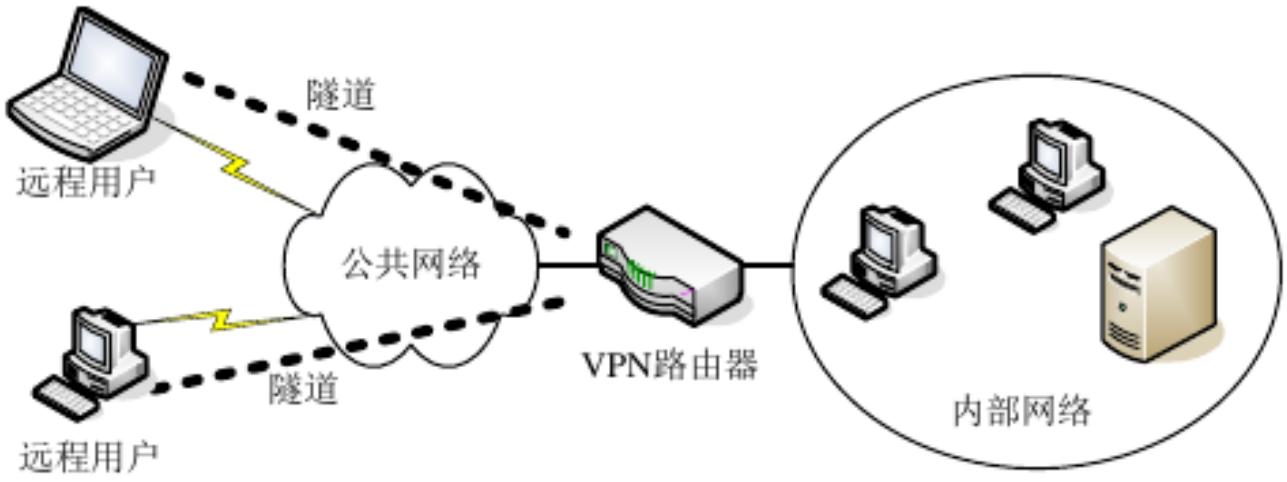
配置动态路由 (4)

- 配置步骤 (3)

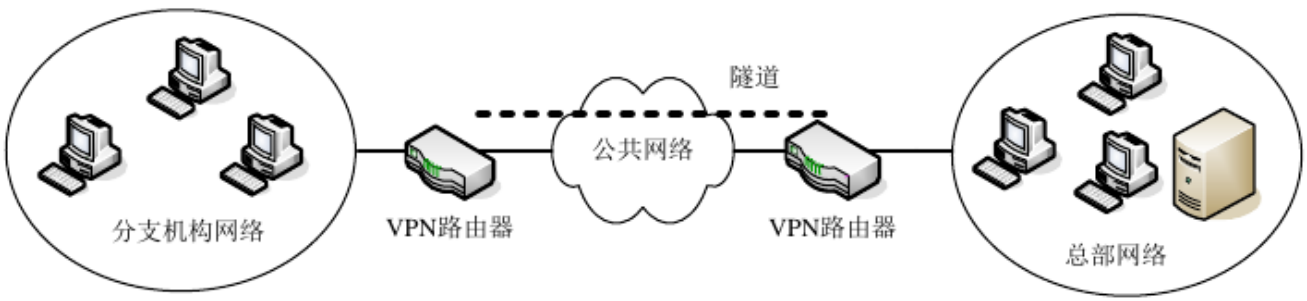
```
service ripd start              # 启动 rip 服务
telnet 127.0.0.1 2601           # 登录 zebra 服务
show ip route
telnet 127.0.0.1 2602           # 登录 rip 服务
enable
show ip rip
```

VPN 概述 (1)

- VPN 应用模式
  - 远程访问



  - 远程网络互联



VPN 概述 (2)

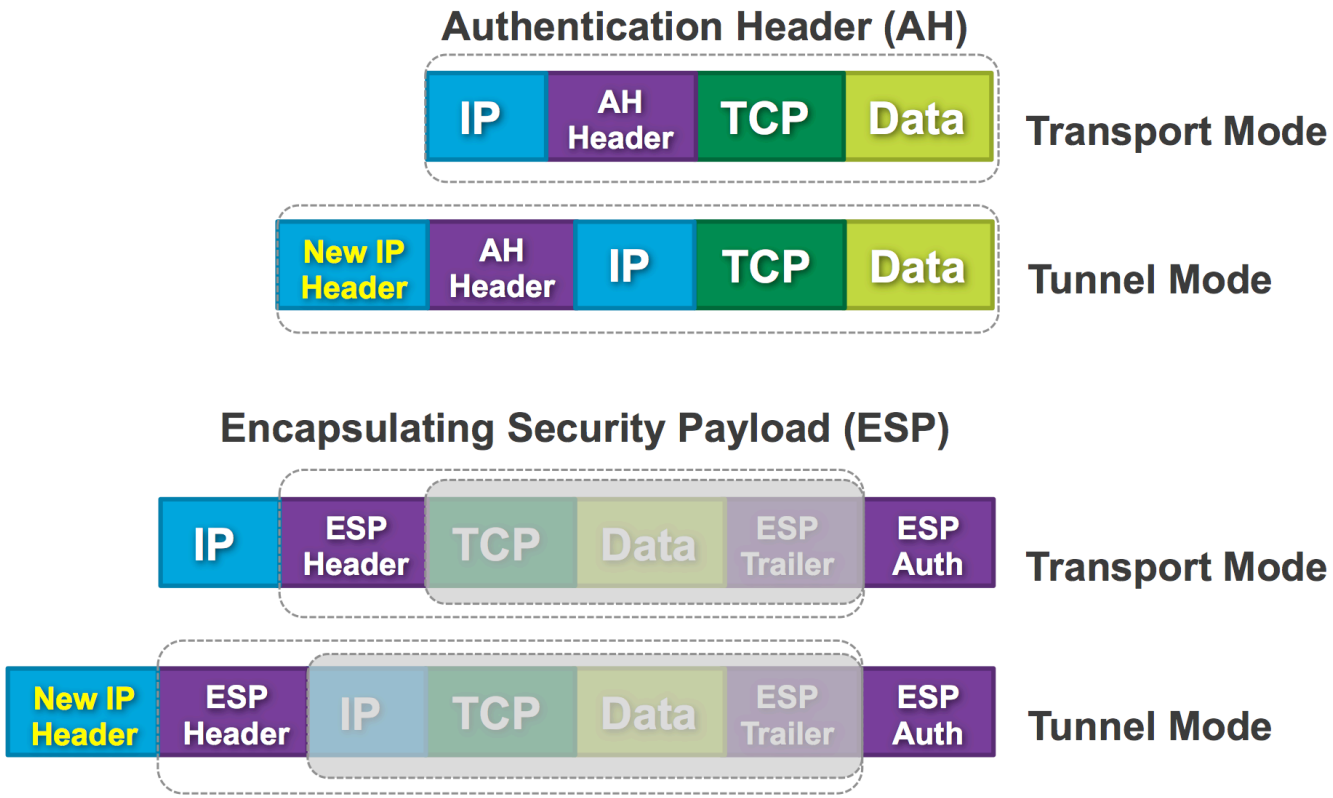
- 基于隧道的 VPN
  - 第二层隧道协议
    - 将链路层协议封装起来进行传输，可在多种网络建立多协议的 VPN，如 PPTP(点对点隧道协议) 和 L2TP(第 2 层隧道协议)。
  - 第三层隧道协议
    - 用于组建 IP VPN，如 IPSec。IPSec 工作在 IP 层，为 IP 层及其上层协议提供保护，对用户和应用程序透明，可为 Internet 业务提供最强的安全功能，适合于组建远程网络互联 VPN。
  - 第四层隧道协议
    - 如 SSL，除具备与 IPSec VPN 相当的安全性外，还增加访问控制机制，客户端只需要拥有支持 SSL 的浏览器即可，适合远程用户访问企业内部网。

ipsec 概述

- IPsec 基于端对端的模式来提供 IP 数据包的安全性，它在源 IP 和目的 IP 地址之间建立信任和安全性。
  - IPsec 工作在网络层 (第 3 层)，一方面可以为 IP 和上层协议提供保护，另一方面对于大多数应用程序、服务和高层协议是透明的。
  - IPsec 使用两个安全协议 AH(Authentication Header, 认证头) 和 ESP(Encapsulating Security Payload, 封装安全载荷)，以及密钥分配的过程和相关协议来实现其目标。
  - AH 可用来保证数据完整性, 提供反重播保护, 并且确保主机的身份验证。ESP 提供和 AH 相似的功能, 另外还提供数据机密性保护。

ipsec 模式

- ipsec 有两种模式：传输模式和隧道模式，AH 和 ESP 均可用于这两种模式。



ipsec 安全协商

- 在两端实现 ipsec 通信之前，必须在某些安全性设置方面达成一致，主要是确定身份验证、完整性和加密算法，这个过程称为安全协商。
- 安全关联 (Security Association, SA)
  - 安全关联存储在 ipsec 两端设备的数据库中，是协商密钥、安全协议与安全参数索引 (spi) 的组合，它们一起定义了用于保护从发送端到接收端的单向安全逻辑连接。
  - 每次 ipsec 通信需要建立一对 SA，一个用于入站通信，一个用于出站通信。每个 SA 使用唯一的 spi 标识。如果一台设备同时与多台设备进行 ipsec 通信，就会存在多个 SA。接收端设备使用 spi 来决定将使用哪个 SA 处理入站数据包。
- Internet 密钥交换 (Internet Key Exchange, IKE)
  - IKE 协议主要有两个作用：
    1. 集中管理安全关联以减少连接时间
    2. 生成和管理密钥

ipsec 安全协商的两个阶段

- 第一阶段
  - 在两端之间建立一个主模式 SA(IKE SA)，是为建立信道而建立的安全关联。这一阶段协商创建一个通信信道，并对该信道进行认证，为双方进一步的 IKE 通信提供机密性、数据完整性以及数据源认证服务。
- 第二阶段
  - 协商一对快速模式 SA(一个 SA 用于入站，一个 SA 用于出站)，是为数据传输而建立的安全关联。这一阶段使用已建立的 IKE SA 协商建立 ipsec SA，为数据交换提供 ipsec 服务。

安全关联数据库和安全策略数据库

- 安全关联数据库 (Security Association Database, SAD)
  - 用于存放 SA 的有关参数：spi 值、目的 IP、AH/ESP、AH 验证算法、AH 验证密钥、ESP 验证算法、ESP 验证密钥、ESP 加密算法、ESP 加密密钥、传输/隧道模式等参数。
- 安全策略数据库 (Security Policy Database, SPD)
  - 用于存放 IPSec 的规则，而这些规则定义哪些流量需要使用 IPSec 进行保护，如: 目的 IP、源 IP、AH/ESP 协议、目的端口、源端口、传输/隧道模式等。

ipsec-tools

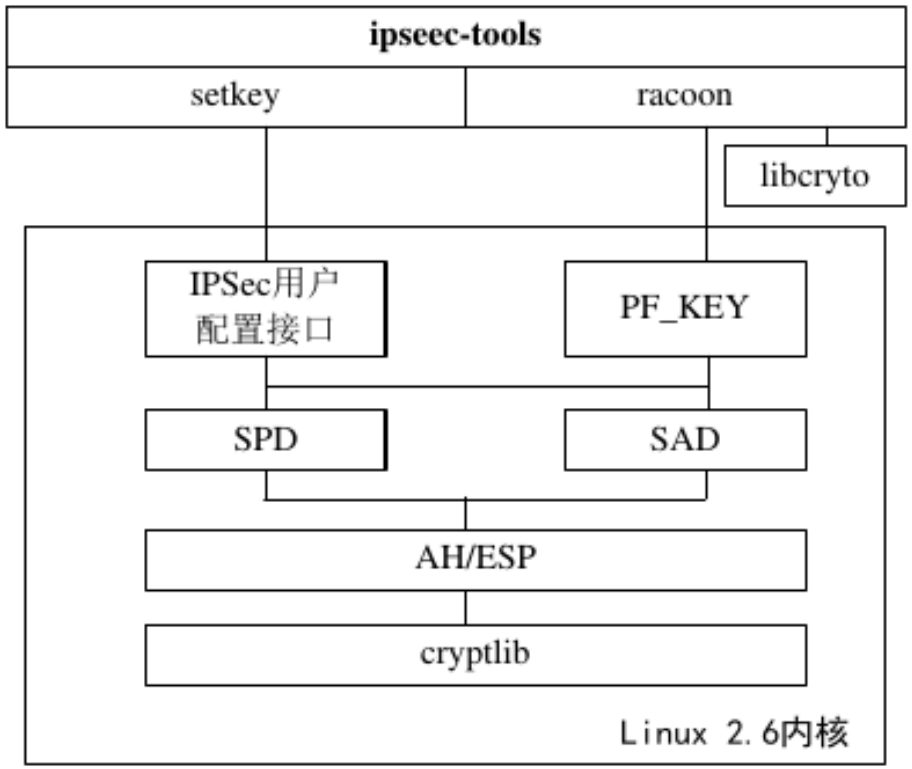
- Linux 传统的 ipsec 解决方案是 FreeS/Wan，但 Linux 内核从 2.6 版开始内置对 ipsec 的支持 (实现了 AH、ESP、SAD、SPD)，并提供 ipsec-tools 工具来配置和管理 ipsec。
- ipsec-tools 包括 setkey 和 racoon 两个程序
  - setkey 用于配置规则策略，实现 SA 和密钥的手动管理

```
setkey -f FILE # 根据 FILE 设置 SAD 和 SPD
setkey -D      # 查看 SAD
setkey -P -D   # 查看 SPD
setkey -F      # 清空 SAD
setkey -P -F   # 情况 SPD
man setkey     # 了解详细信息
```

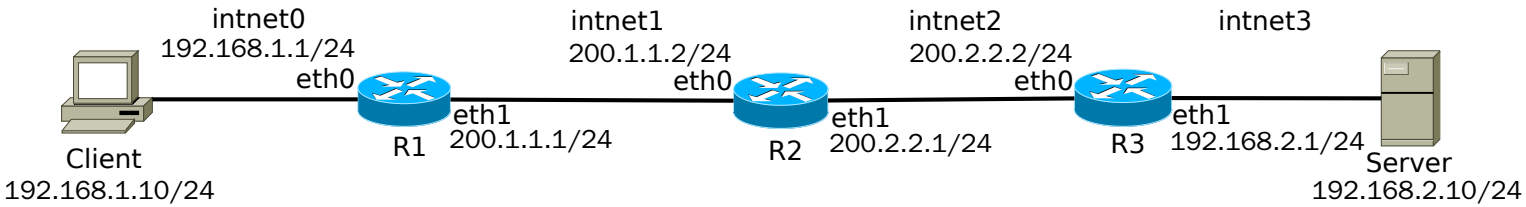
  - racoon 用于密钥协商，是一种 IKE 实现，实现 SA 和密钥的自动管理

ipsec-tools 数据处理模型

- 数据处理过程
  对进出数据首先确定其安全策略，如果需要安全服务，则找到对应的 SA，根据 SA 相关参数进行 AH/ESP 封装后再完成数据的输入/输出。



配置主机到主机的 ipsec 连接 (1)



- 配置步骤（以 Client 端为例，Server 端类似）
  - 1. 编辑 ipsec 接口配置文件

```
vim /etc/sysconfig/network-scripts/ifcfg-ipsec0
DST=192.168.2.10 # 对端主机 IP 地址
TYPE=IPSEC       # 接口类型
ONBOOT=no        # 系统启动时是否激活
IKE_METHOD=PSK   #IKE 验证方式（此处采用预共享密钥）
```

  - 2. 编辑预共享密钥文件

```
vim /etc/sysconfig/network-scripts/keys-ipsec0
IKE_PSK=Abc_123
# 为保障密码安全，最好修改 keys-ipsec0 的权限
chmod 600 /etc/sysconfig/network-scripts/keys-ipsec0
```

配置主机到主机的 ipsec 连接 (2)

- 配置步骤 (以 Client 端为例，Server 端类似)(续)
  - 3. 重启网络服务

    ```
    service network restart
    ```

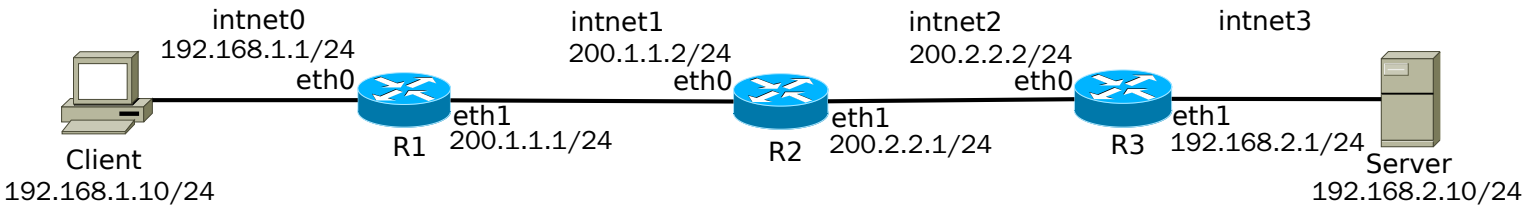
  - 4. Server 端类似执行 1 ～ 3 步
  - 5. 在 Client 和 Server 上激活 ipsec0 接口

    ```
    ifup ipsec0 # 将自动生成/etc/raccoon/192.168.2.10.conf
    ```

  - 6. 测试并用 tcpdump 抓包验证

    ```
    client: ping -c 4 192.168.2.10
    server: tcpdump -i eth0
    ```

配置网络到网络的 ipsec 连接 (1)



- 配置步骤 (以 R1 为例，R3 类似)
  - 1. 编辑 ipsec 接口配置文件

```
vim /etc/sysconfig/network-scripts/ifcfg-ipsec1
TYPE=IPSEC
ONBOOT=yes
IKE_METHOD=PSK
SRCGW=192.168.1.1          # 源网关地址
DSTGW=192.168.2.1          # 目的网关地址
SRCNET=192.168.1.0/24      # 源网络
DSTNET=192.168.2.0/24      # 目的网络
DST=200.2.2.2              # 隧道对端公网地址
```

配置网络到网络的 ipsec 连接 (2)

- 配置步骤 (以 R1 为例，R3 类似)(续)
  - 2. 编辑预共享密钥文件

```
vim /etc/sysconfig/network-scripts/keys-ipsec0
IKE_PSK=Abc_123
chmod 600 /etc/sysconfig/network-scripts/keys-ipsec0
```

  - 3. 确保 R1 启用了 ip 转发功能

```
sysctl -a | grep ip_forward
```

  - 4. 在 R3 上类似执行 1 ～ 3 步
  - 5. 在 R1 和 R3 上激活 ipsec1 接口

```
ifup ipsec1 # 将自动生成/etc/raccoon/200.2.2.2.conf
```

  - 6. 测试并用 tcpdump 抓包验证

```
client: ping -c 4 192.168.2.10
    R3: tcpdump -i eth0
server: tcpdump -i eth0
```

服务的类型

- 按功能分
  - 系统服务: 指那些为系统本身或者系统用户提供的一类服务, 如提供作业调度服务的 cron 服务。
  - 网络服务: 网络服务是指供客户端调用的一类服务, 如 Web 服务、文件服务等。
    - 网络服务定义文件：/etc/services
- 按启动方式分
  - 独立服务 (Standalone Service): 启动后始终在后台执行, 除非关闭系统或强制中止。多数服务属于此种类型。
  - 临时服务 (Transient Service): 只有当客户端需要时才会被启动, 使用完毕就会结束。
    - 超级服务 xinetd 用于管理其他临时服务

服务控制脚本

- 以 rpm 包方式安装的服务的控制脚本统一放置在目录/etc/rc.d/init.d 目录
- 利用服务控制脚本控制服务

```
/etc/rc.d/init.d/sshd \
start|stop|restart|reload|condrestart|status
reload       # 在不重新启动服务的前提下重新加载其配置文件
condrestart # 只有服务正在运行时才重新启动该服务
service sshd start|stop|...
```

配置服务器是否自动启动

- 以 rpm 包方式安装的服务可用 chkconfig 或 ntsysv 命令配置其是否启动
  - chkconfig 命令

```
chkconfig --list       # 查看所有服务的启动配置
chkconfig --list sshd  # 查看 sshd 服务的启动配置
chkconfig --level 234 sshd on  # 运行级别 234 启动 sshd
chkconfig --level 24 sshd off  # 运行级别 24 关闭 sshd
```

  - ntsysv 命令

```
ntsysv                 # 对当前运行级别下的服务进行启动配置
ntsysv --level 235     # 对运行级别 235 下的服务进行启动配置
setup                  # 可以通过 setup 命令对服务进行启动配置
```

PAM

- 为了解决多个应用的用户认证问题，Linux 引入了一套名为可插拔认证模块 (Pluggable Authentication Modules, PAM) 的用户认证机制，将用户认证功能从应用中独立出来，单独进行模块化设计，统一实现和维护，并提供了一套标准 API，以便各应用程序能够方便地使用它们所提供的各种功能。
- PAM 为了提供足够高的通用性、灵活性和可配置性，采用了插件机制，并采用了分层的体系结构。

PAM 认证机制 (1)

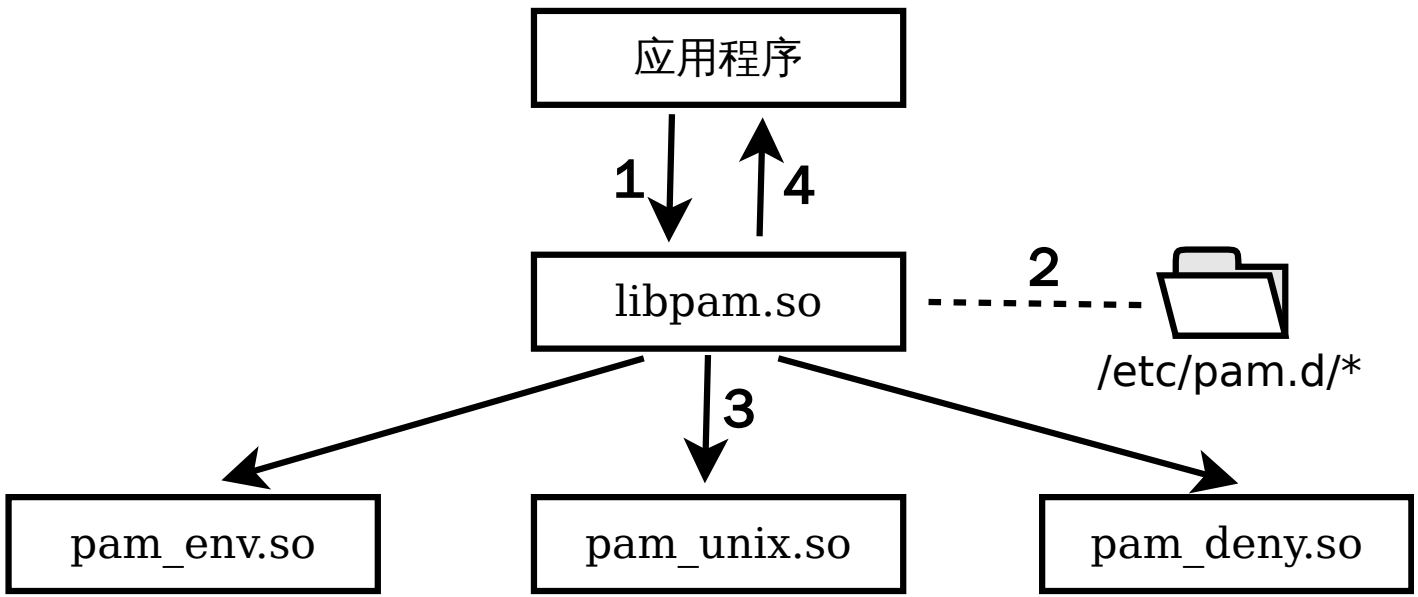
- PAM 认证过程
  1. 采用 PAM 认证的服务或应用程序向 PAM 库提出验证请求，PAM 实际上是一个名为 libpam.so 的共享链接库。

```
ldd /bin/login | grep libpam.so # 查看应用是否包含 PAM
```

  1. PAM 库确定提出请求的是哪一项服务或应用程序，然后到/etc/pam.d 目录下加载相应服务或应用程序的 PAM 的配置文件 (与服务或应用程序同名，如 vsftpd 服务的配置文件为/etc/pam.d/vsftpd)。
  2. PAM 库根据 PAM 配置文件的设置调用指定的认证模块 (位于/lib/security 目录下) 进行安全认证。
  3. PAM 库将认证结果返回给服务或应用程序。

PAM 认证机制 (2)

- PAM 体系架构



PAM 认证机制 (3)

- PAM 客户端可以请求 PAM 内核执行下列动作：
  - 确认用户身份
  - 取得账户信息
  - 修改验证的凭证或帐号数据
- 每个 PAM 模块执行后都会返回 success 或 fail，PAM 内核 (libpam.so) 再根据该结果及 PAM 客户端配置文件的配置决定下一步动作。当 PAM 内核执行完所需执行的模块后，把最终结果返回给 PAM 客户端，PAM 客户端则由该结果判断验证过程是否成功。
- 注意：如果 PAM 找不到指定模块，或者 PAM 模块无法顺利执行，则 PAM 内核和 PAM 模块的默认返回结果都是 fail。

PAM 客户端配置文件 (1)

- 每个支持 PAM 的应用程序或服务都在/etc/pam.d 目录中有一个相应的配置文件，其文件名与相应的应用程序或服务同名。
- PAM 客户端配置文件格式

  类型 控制标记 模块路径 [模块参数]

- PAM 认证过程根据客户端配置文件内容顺序执行。

PAM 客户端配置文件 (2)

- 类型
  - auth
    认证管理，对用户的身份进行识别，如提示用户输入密码。
  - account
    账户管理，对账户各项属性进行检查，如是否允许登录，限制最大用户数。
  - password
    密码管理，更新密码和凭证等，如修改用户密码。
  - session
    会话管理，定义用户登录前和退出后要进行的操作，如显示登录连接信息。

PAM 客户端配置文件 (3)

- 控制标记 (1)
  - required

    表示该模块认证成功是用户通过认证的必要条件。只要模块认证失败，用户就一定不会通过认证。但即便失败，PAM 也不会立即将结果返回给应用，而是继续调用其他模块。这样做的目的就是为了麻痹“敌人”，让他们搞不清楚到底是哪个环节认证失败的。
  - requisite

    与 required 类似，但是只要 requisite 模块认证失败，就会立即返回给应用。一般用于判定当前用户所处的环境，如果环境不够安全，即便是合法的用户也不会通过验证。这是最严苛的要求，一般很少使用。但是 requisite 能够将低黑客利用不安全媒介获得输入密码的机会。

PAM 客户端配置文件 (4)

- 控制标记 (2)
  - sufficient
    表示该模块验证成功是用户通过认证的充分条件。只要这个模块验证成功了，就代表没有必要继续去认证其他模块了，将立即返回给应用。但是其优先级低于 required，如果其前面有 required 模块失败，则最终结果也是失败，当 sufficient 认证失败时，相当于 optional。
  - optional
    表示即便该模块认证失败，用户也可能通过认证，除非别无它选。实际应用中，optional 模块只是显示相关信息，根本不做认证工作。
  - include
    执行指定 PAM 客户端配置文件中与类型字段匹配的那些模块

PAM 客户端配置文件 (5)

- system-auth 配置文件
  - 几乎每一个 PAM 客户端都会包含以下设置

```
auth       include system-auth
account    include system-auth
password   include system-auth
session    include system-auth
```

  - 因为许多应用程序会利用 PAM 进行验证授权，且要执行的动作几乎一样，system-auth 为此提供了一个全局默认配置，如果需要修改所有 PAM 客户端默认的验证授权配置，只需修改 system-auth 文件，无需逐个修改每一个 PAM 配置文件。

```
cat /etc/pam.d/system-auth
```

常用 PAM 模块 (1)

- pam_unix.so 模块 (1)

  该模块利用名称服务切换器取得帐号数据，然后用帐号数据进行验证授权动作。
  - pam_unix.so 模块用于不同类型时执行不同的动作
    - auth 以 NSS 解析所得的密码进行身分验证
    - account 以 NSS 解析所得的密码数据判断密码与帐号是否过期
    - password 将修改后的密码存储至本机或 NIS 服务器
    - session 记录登录、注销信息

常用 PAM 模块 (2)

- pam_unix.so 模块 (2)
  - 常用参数
    - debug 产生除错信息
    - use 如果同时使用多种验证方法，则以第一顺位的为准
    - try 除非没有设置 PAM，否则不提示用户输入密码
    - use 与 try 相同，但没有设置 PAM 时会返回失败
    - shadow 从 shadow 中取得密码数据
    - md5 取的密码数据是经过 MD5 加密后的密码
    - nis 通过 NIS 服务器取得帐号数据
    - remember=N 记忆过去 N 次内的密码，即新设置的密码不得与过去 N 次的密码相同

常用 PAM 模块 (3)

- pam_deny.so 模块
  - 直接返回失败，可应用于任何类型
- pam_permit.so 模块
  - 直接返回成功，仅应用于 account 类型
- pam_rootok.so 模块
  - 执行时判断用户的 UID 是否为 0(root 用户)，是则返回成功，否则返回失败

常用 PAM 模块 (4)

- pam_cracklib.so 模块
  - 检查密码强度，仅应用于 password 类型
  - 常用参数
    - debug 显示除错信息
    - type=TEXT 修改密码时以 New TEXT password 作为提示串
    - retry=N 发生错误时，提示用户的最大次数
    - difok=N 与旧密码相同字符的最大数量
    - minlen=N 密码最少字符数
    - dcredit=N 密码中至少包含 N 个数字
    - ucredit=N 密码中至少包含 N 个大写字符
    - lcredit=N 密码中至少包含 N 个小写字符
    - ocredit=N 密码中至少包含 N 个例如 @、#、～等的其他字符

常用 PAM 模块 (5)

- pam_limits.so 模块
  - 根据指定的配置文件限制 PAM 客户端能使用多少系统资源，如 CPU 运算能力、启动进程数、最大登录次数等，仅应用于 account 类型。
  - 常用参数
    - debug 显示除错信息
    - conf=FILE 以 FILE 作为配置文件，如未缺省，则默认配置文件为/etc/security/limits.conf

常用 PAM 模块 (6)

- pam_access.so 模块
  - 根据用户登录位置决定可否执行 PAM 客户端，通常用于 account 类型。
  - 常用参数
    - accessfile=FILE 指定配置文件，如果缺省，则默认配置文件为/etc/security/access.conf
    - fieldsep=SEP 指定字段间分隔符，默认为冒号 (:)
    - listsep=SEPs 如需指定多笔数据，则用来指定每一笔数据之间的分隔符，默认为空格或制表符
    - access.conf 的配置格式

```
man 5 access.conf
```

常用 PAM 模块 (7)

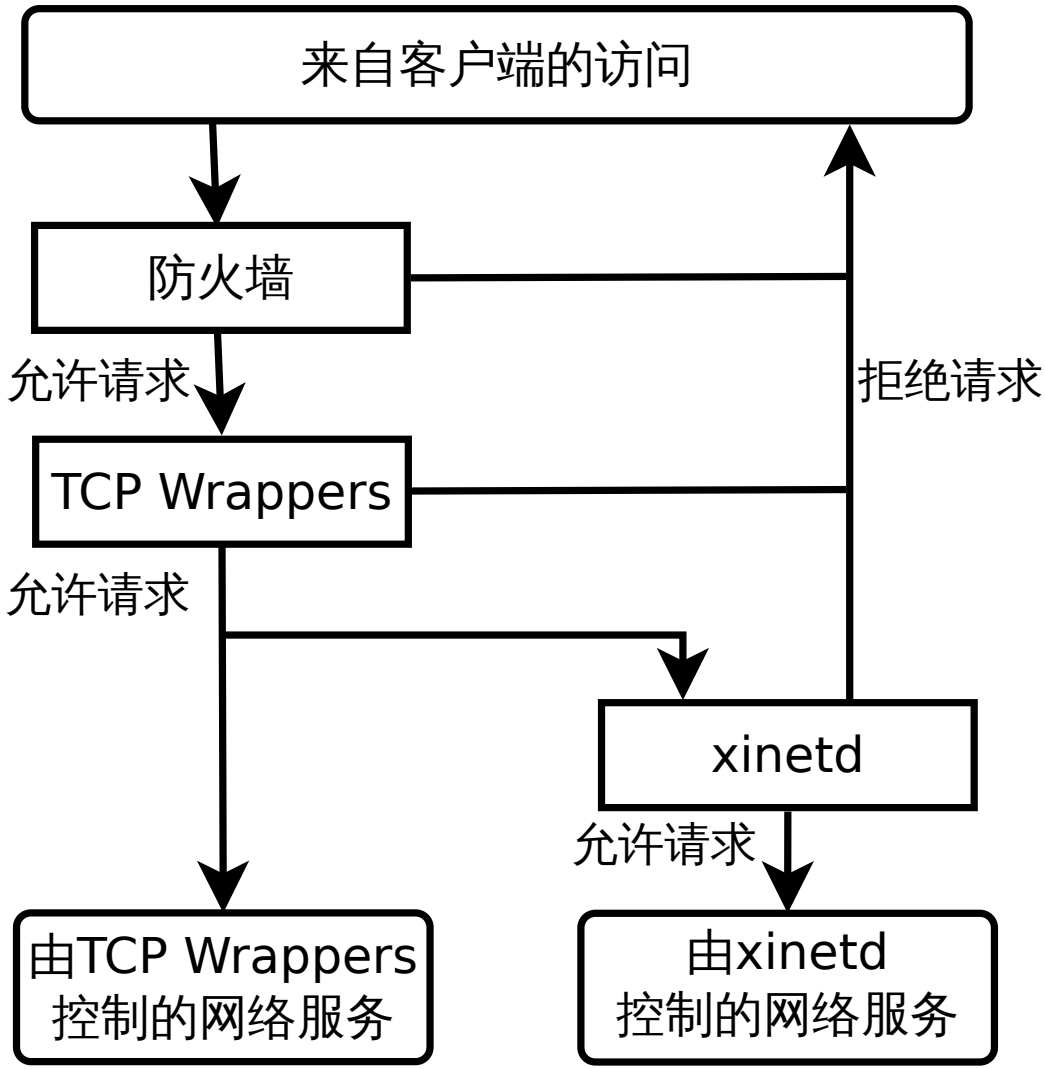
- pam_tally.so 模块
  - 计算用户登录失败的次数，一旦成功则归零。
  - 常用参数
    - file=File 指定用于保存登录失败次数的记录文件，默认为/var/log/faillog
    - onerr=RESULT 如果找不到记录文件或发生异常状况，则默认返回何种结果？可以是 success 或 fail。
    - deny=N 登录失败次数达 N 次，就返回失败，用于 auth 类型
    - no 即使用户登录成功也不把登录失败次数归零。
- 更多模块的详细介绍，可查看 pam 文档

```
rpm -qd pam
```

概述

- Linux 提供 TCP Wrappers 工具为其增加一个保护层用于控制主机到网络服务的连接, 大多数网络服务都可利用 TCP Wrappers 在客户端请求与服务之间建立防护机制。
- xinetd 是一种超级服务器, 本身也是一种由 TCPWrappers 管控的服务, 可以控制对一个网络服务子集的访问。
- 可以组合使用 iptables 防火墙、TCP Wrappers 和 xinetd 实现网络服务多重访问控制。

网络服务多重访问控制


来自客户端的访问
防火墙
允许请求
拒绝请求
TCP Wrappers
允许请求
xinetd
允许请求
由TCP Wrappers
控制的网络服务
由xinetd
控制的网络服务

TCP Wrappers(1)

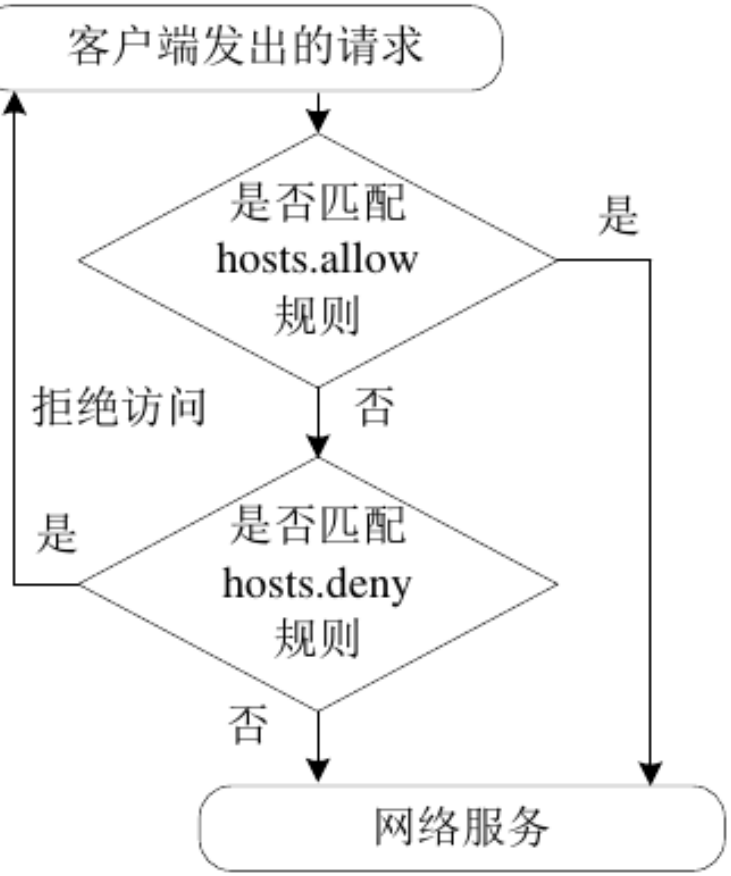
- 要开放一个本地系统以从远程系统访问时应遵循的原则：
  1. 只对你允许访问的系统开放本地系统
  2. 只允许每个远程系统访问你让它访问的数据
  3. 只允许每个远程系统以适当的方式 (只读、读写、只写) 访问数据
- TCP Wrapper 以/etc/hosts.allow 和/etc/hosts.deny 文件控制对系统网络服务的访问，可用于链接到 libwrap 库的任何守护进程。

```
ldd /usr/sbin/sshd | grep libwrap # 查看 sshd 是否链接
```

- 由 TCP Wrappers 控制的服务并不缓存主机访问文件中的规则，因此对 hosts.allow 和 hosts.deny 的配置修改立即生效，无需重启系统或网络服务。

TCP Wrappers(2)

- hosts.allow 和 hosts.deny 文件格式
  - 服务列表: 客户列表 [: 命令]
- 客户端请求访问某个服务时，按照下列顺序进行访问控制：
  1. 如果服务进程/客户端对匹配 hosts.allow 中的一行，允许访问
  2. 否则，如果服务进程/客户端对匹配 hosts.deny 中一行，拒绝访问
  3. 否则，如果在 hosts.allow 和 hosts.deny 中均无匹配，允许访问



TCP Wrappers(3)

- 示例：

/etc/hosts.deny

```
ALL:ALL:echo '%c try to connect %d - blocked' \
>>/var/log/tcpwrappers.log  # 默认拒绝所有访问
```

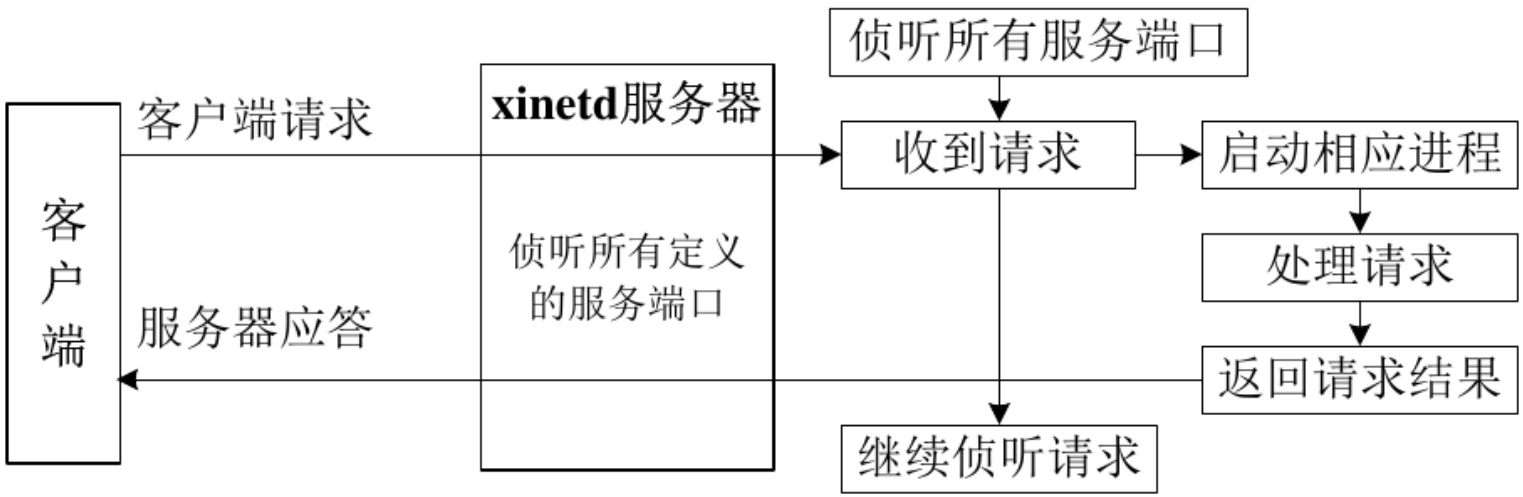
/etc/hosts.allow

```
sshd:ALL              # 允许从任何系统连接 sshd
in.telnet:LOCAL   # 允许从本域连接 telnet
in.telnet:192.168.56.* 127.0.0.1 # 允许指定 ip 连接 telnet
# 允许 172.16 网段中除 172.16.251.105 之外所有主机访问 telne
in.telnetd: 172.16. EXCEPT 172.16.251.105
# 仅允许 192.168.56.1 访问 telnet 服务
in.telnetd:ALL EXCEPT 192.168.56.1:deny
```

xinetd 超级服务 (1)

- xinetd 本身是一个独立运行的服务，它能够同时监听多个指定的端口，在接受用户请求时，能够根据用户请求的端口不同，启动不同的网络服务进程来处理这些用户请求。
  - 可以将 xinetd 看作是一个管理启动服务的服务器，它决定将一个客户请求交给哪个程序处理，然后启动相应的进程，完成服务请求后，再结束该进程的运行，以减少对系统资源的占用。
- 理论上任何服务都可以使用 xinetd。但是，对于访问量大、经常出现并发访问的服务，xinetd 要频繁启动对应的进程，反而会影响系统性能。因此，xinetd 主要用于管理系统中不频繁使用的服务。

xinetd 超级服务 (2)



xinetd 超级服务 (3)

- 由于 xinetd 本身是由 TCP Wrappers 控制的服务，因而对其管理的服务进行访问控制时，会同时用 TCP Wrappers 和 xinetd 控制机制。
- 连接由 xinetd 控制的一个网络服务时，会按照以下步骤实现访问控制：
  1. xinetd 守护进程通过调用 libwrap.a 库来检查 TCP Wrappers 主机访问规则。
  2. xinetd 守护进程检查其本身的访问控制规则，包括 xinetd 服务和被请求的服务的控制规则。
  3. 连接建立后，xinetd 就不再参与客户端和服务器之间的通信。

xinetd 超级服务 (4)

- 全局配置文件/etc/xinetd.conf
  - 全局配置文件/etc/xinetd.conf 包含一般的设置，这些设置影响 xinetd 所控制的每一项服务。
  - xinetd 服务第一次启动时读取该文件设置信息，所以要想使修改的配置起作用，还需要重新启动 xinetd 服务。
- 特定服务配置目录/etc/xinetd.d/
  - xinetd 将所控制的每项服务的配置都储存在一个独立的文件中，这样更便于管理员分别定制。
  - 每项服务的配置文件位于/etc/xinetd.d/目录中，以服务的名称命名，文件格式与/etc/xinetd.conf 相同。