

Chapter 1. 搭建服务器前的准备工作

1.1. Linux 的功能

- 1、误区：学 Linux 就是为了搭建服务器，就不用学习其他功能，例如计划任务，Bash Shell 等等功能
- 2、任何一个曾经有过搭建发布到 Internet 上的网站的朋友来说，上面这些话真会害死人

1.1.1. 用 Linux 搭建服务器需要的能力

- 1、说穿了，Linux 就是一套非常稳定的操作系统，任何工作只要能在 Linux 这个操作系统上面运行，那它就是 Linux 可实现的功能之一
- 2、对于一个服务器而言“搭建容器维护难，除错更难”

1.1.2. 搭建服务器难不难呢

- 1、搭建一台堪称完美的服务器，基本功课包括：
 - 网络的基本概念，以方便进行联网与配置及排错
 - 熟悉操作系统的基本操作：包括登录控制、账号管理、文本编辑器的使用技巧等
 - 信息安全方面：包括防火墙与软件更新的相关知识
 - 该服务器协议所需软件的基本安装、配置、排错等

1.2. 搭建服务器的基本流程

1.2.1. 网络服务器成功连接的分析

- 1、了解操作系统的基础对于服务器的维护是相当重要的
- 2、你连接到服务器，重点在取得服务器上的数据，而一般数据的存在就是使用文件
- 3、流程如，见下图

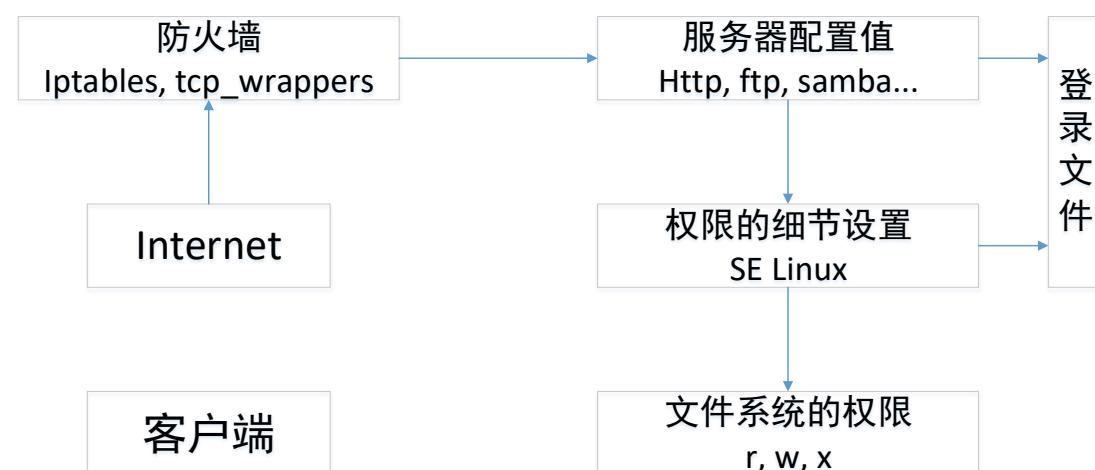


图 1-1 通过网络连接至服务器所需经过的各项环节

- 首先客户端到服务器的网络要能够连接
- 等到客户端访问到服务器后，会先由服务器的防火墙判断该连接是否能放

行

- 等到放行之后才能使用到服务器软件的功能
- 而该功能又需要通过 SE Linux 这个细节权限设置的项目后，才能够读取到文件系统
- 是否能够读取文件，又与文件系统的权限 r,w,x 有关

4、根据上面的流程大致可以将整个连接分为几个部分，包括：网络、服务器本身、内部防火墙设置，各项服务配置文件、细节权限的 SE Linux 以及最重要的文件权限

- **网络：**了解网络基础知识与所需服务的通信协议
 - 基本的网络基础知识：包括以太网络硬件与协议、TCP\IP、网络连接所需参数等
 - 各网络服务所对应的通信协议的工作原理，以及实现各通信协议的具体应用程序
- **服务器本身：**了解搭建网络服务器的目的以配合主机的安装规划
 - 搭建什么样的服务器
 - 该服务器要不要对 Internet 开放
 - 这个服务器要不要对客户提供访问账号
 - 要不要针对不同的访问账号进行，例如磁盘容量，可用空间与可用系统资源的限制等
- **服务器本身：**了解操作系统的基本操作
 - 包括软件如何安装与删除，如何管理系统的计划任务，如何根据服务器的服务目的规划文件系统
 - 如何让文件系统具有可扩展性(LVM)
 - 系统如何管理各项服务的启动
 - 系统的开机流程是什么
 - 系统出错时，该如何进行快速复原等
- **内部防火墙设置：**管理系统的可共享资源
 - 在了解网络基础与所需服务的预期目的之后，通过防火墙来规范可以使用户，以让系统在使用上拥有较佳的可控环境
 - **不管你的防火墙系统设置的再怎么严格，只要是你开放的服务，那防火墙对于该服务就没有保护的效果**
 - 在线更新软件机制一定要定期进行，否则系统将会非常不安全
- **服务器软件设置：**学习设置技巧与开机是否进行自动执行
 - 软件如何安装；如何查询相关配置文件的所在位置
 - 服务器软件如何设置
 - 服务器软件如何启动；如何设置自动开机启动；如何观察启动的端口
 - 服务器软件激活失败如何排错；如何查看日志；如何通过日志进行除错
 - 通过客户端进行连接测试，如果失败该如何处理？连接失败的原因是服务器还是防火墙
 - 服务器的设置修改是否有相关的日志；相关日志是否要定期分析
 - 服务器所提供货共享的数据有无定期备份；如何定期自动备份或远程备份
- **细节权限设置：**包括 SE Linux 与文件权限

1.2.2. 一个常见的服务器设置案例分析

- 1、网络环境：假设你的环境里面有 5 台计算机，这 5 台计算机需要通过网络连接在一起，且都可以对外提供连接访问
- 2、对外网络：你的环境只有一个对外的连接，这里假设是 ADSL 或 10M 的光纤
- 3、额外服务：你要让这 5 台计算机都可以上网，而且其中还有一台主机可以作为文件服务器，作为数据备份与数据共享之用
- 4、服务器管理：由于你可能需要进行远程管理，因此你这台服务器需要开放连接机制，以让远程计算机可以连接到这部主机来进行维护
- 5、防火墙管理：因为担心作为文件共享服务器的系统被攻击，因此你需要针对源 IP 来进行登录控制
- 6、账号管理：由于同学的数据有隐私与共享之分，因此你还需要为每个同学提供专门的访问账号，且每个账号都有磁盘容量的使用限制
- 7、后台分析：由于担心系统出问题，所以你需要让系统自动定期分析磁盘的使用量、日志文件参数信息等

1.2.2.1. 了解网络基础

1、硬件规划

- 想要将 5 台计算机连接在一起，却又只有一个可以对外提供连接，此时就需要购买集线器(Hub)或者是交换器(Switch)来连接所有计算机

2、连接规划

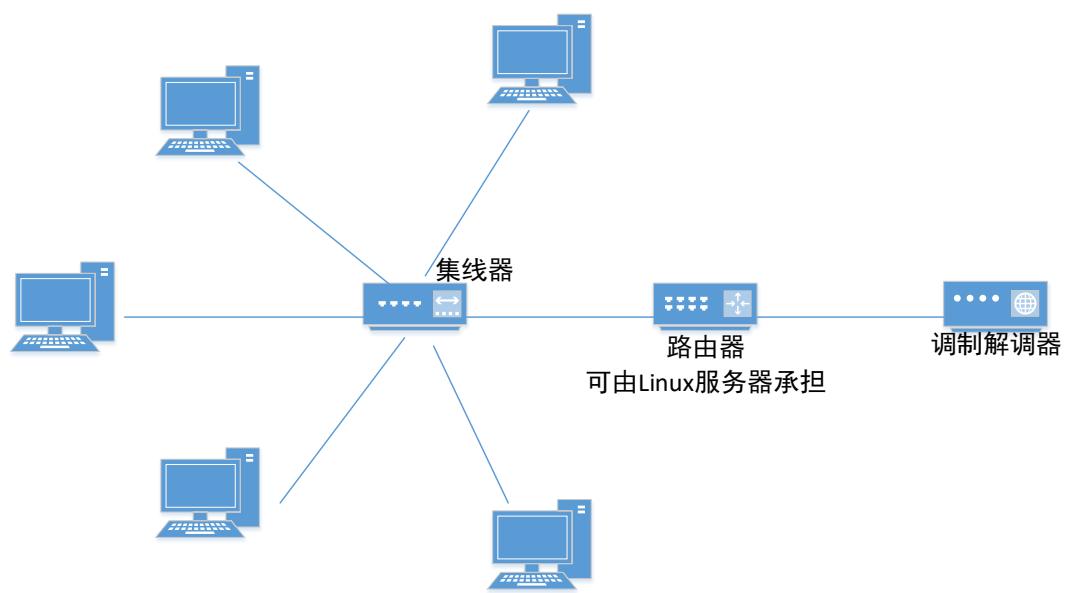


图 1-2 硬件的网络连接示意图

- 通过路由器，5 台计算机就都能够上网了，能否上网与 Internet 有关，其核心就是著名的 TCP\IP 通信协议

3、网络基础

- 这部分比较复杂，包括 TCP\IP、Network IP、Netmask IP、Broadcast IP、Gateway、DNS(Domain Name System) IP 等等

1.2.2.2. 服务器本身的安装与服务器目的的搭配

- 1、如图 1-2 所示，Server 端是在五台计算机之中，而且 Server 必须要针对不同

账号分配磁盘空间，我们这里会提供共享(SAMBA)这个服务，该服务可以在 Windows 与 Linux 之间通用，由于要提供账号给用户，以及想到未来磁盘扩展的情况，因此我们想要将/home 独立出来，且使用 LVM 这个管理模式，并搭配 Quota 机制来控制每个账号的磁盘使用情况

2、Quota 仅支持文件系统而不支持单一目录，这是/home 需要单独分区的原因之一

1.2.2.3. 服务器操作系统的基本使用

1、<未完成>

1.2.2.4. 服务器内部的资源管理与防火墙规划

- 1、查看服务的启动状态：`systemctl list-unit-files --type=service`
- 2、查询已启动的网络监听服务：`netstat -tulnp`
- 3、对外开放的软件没有更新，那么防火墙形同虚设，所以软件更新很重要
- 4、利用 yum 进行系统更新
 - 假设网络已经通了，目前想要进行整个系统的更新，同时需要每天凌晨 2:15 自动进行整个系统的更新
 - `vim /etc/crontab`
 - `15 2 * * * root /usr/bin/yum -y update`
 - 如果系统曾经更新过内核(kernel)这个软件，务必要重新启动，因为内核是在开机时加载的，一经载入就无法在这次的操作中更改版本

1.2.2.5. 服务器软件设置：学习设置技巧与开机是否自动执行

- 1、若想要提供一个网络文件服务器，那么网络文件服务器使用的机制有哪些呢
 - 常见的除了网页形式的共享磁盘外，还有常见的网上邻居以及 Linux 的 NFS 方式
 - 假设局域网内的操作系统大部分是 Windows，因此网上邻居应该是个比较合理的磁盘共享机制
 - 网上邻居究竟启动了几个端口？
 - 如何持续提供网上邻居数据？
 - 访问的账号有没有限制？
 - 访问的权限如何设置？
 - 是否可以规定谁可以登录某些特定的目录？
 - 针对网上邻居服务的端口该如何设置防火墙？
 - 如果系统出错该如何查询错误？
 - 网上邻居的实现在 Linux 环境中是由 Samba 这套软件来完成的

2、搭建一个网上邻居服务器，应该要掌握的基础知识有哪些，以及理论上应该要经过哪些步骤与过程

- 软件安装与查询
 - `rpm -qa | grep -i samba` <=在已安装的软件中查找
 - `yum install samba` <=安装 samba
 - `rpm -qc samba samba-common` <=查询 samba 与 samba-common 的配置文件所在的目录
- 服务器的基本配置与相关配置

- 你必须清楚地知道，你到底需要的服务是什么，针对该服务需要设置的项目有哪些，这些设置需要用到什么命令或配置文件等
- 一般来说，需要先查看这个服务使用的通信协议是什么，然后了解如何设置，接下来编辑主配置文件，根据主配置文件的数据去执行相对应的命令来取得正确的环境设置
- 以网上邻居为例
 - ◆ 先使用 vim 编辑 /etc/samba/smb.conf 配置文件
 - ◆ 利用 useradd 建立所需要的网上邻居实体用户
 - ◆ 利用 smbpasswd 建立可用网上邻居的实体账户
 - ◆ 利用 testparm 测试一下所有数据语法是否正确
 - ◆ 检查看看在网上邻居共享的目录权限是否正确
- 服务器的启动与观察
- 客户端的连接测试
- 错误的解决与查看日志文件
 - 先看看相关日志文件有没有错误信息，举例来说，Samba 除了会在 /var/log/messages 里面列出相关信息外，大部分的日志信息应该放置在 /var/log/samba/ 这个目录下
 - 将信息带入 Google 查询，通常可以解决日志中出现的，但你没有办法解决的问题，达标率可达 95% 以上
 - 如果还是不成功，就到各大讨论区发问，建议到酷学园(phorum.study-area.org)
 - 最常见的错误其实是 SELinux 的错误，此时就需要使用 SELinux 的方法来尝试处理了

3、总结：搭建一台主机需要了解的内容

- 各个 process 与 signal 的观念
- 账号与组的概念与相关性
- 文件与目录的权限，这当然包含于账号相关的特性
- 软件管理的学习
- Bash 的语法与 Shell Scripts 的语法，还有 vim
- 开机的流程分析，以及日志文件的设置与分析
- 类似 Quota 以及文件系统连接等概念

1.2.2.6. 详细权限与 SE Linux

1、特殊情况：假设有一个 student 账号，现在 vbirdgroup 的组想要让 student 这个用户可以进入该共享目录查阅，但是不能改变原本的数据

- 传统的身份与权限的解决方案
 - 让 student 加入 vbirdgroup 组群(usermod student -G vbirdgroup)，但如此一来，student 具有 vbirdgroup 的 rwx 权限，也就可以写入与修改，因此不行
 - 将 /home/vbirdgroup 的权限改为 2775 即可，如此一来，student 拥有其他人的权限，但所有其他人均拥有 rx 权限了，这个方案也行不通
- 使用 ACL
 - 让 student 可以进入 /home/vbirdgroup 进行查询，但不可以写入，同时 vbirduser5

- setfacl -m u:student:rx /home/vbirdgroup
- setfacl -m u:vbirduser5:- /home/vbirdgroup

2、SE Linux

- SE Linux 主要在控制特殊的权限，它可以针对某些程序要读取的文件来设计 SE Linux 类别，当程序与文件的类别形态可以相符合时，该文件才能开始被读取
- 即便配置文件被设置为 777，但是因为程序与文件的 SE Linux 不符合，因此程序还是无法读取该文件的，见图 1-1，SE Linux 绘制在 daemon 与 file permission 中间
- SE Linux 是很复杂的，但是如果仅想应用而已，那么 SE Linux 的处理方式全部可以通过日志来处理。SE Linux 出现问题的机会非常大，但是解决技巧却很简单，就是通过日志内的说明去做即可

1.2.3. 系统安全与备份处理

1、硬件问题比操作系统与软件问题还要严重，而人为的问题比硬件问题严重

2、规范严格的密码规则

- 修改/etc/login.defs 文件里面的规则，让用户需要每半年更改一次密码，且必须大于 8 个字符
- 利用/etc/security/limits.conf 来规范每个用户的相关权限
- 利用 pam 模块来额外的进行密码验证工作

3、虽然防火墙无用论常常被提及，但是 netfilter(Linux 的核心内置防火墙)其实仍有它存在的必要。因此你还是需要根据你自己的主机环境来设计专属于自己的防火墙规则，例如上面提到的 SSH 服务中，你可以仅针对某个局域网络或某个特定 IP 开放连接功能即可

4、备份是很重要的一环，可以用 Shell Script 来定期备份

```
#!/bin/bash
backdir="/etc /home /root /var/spool/mail"
basedir=/backup
[ ! -d "$basedir" ] && mkdir $basedir
backfile=$basedir/backup.tar.gz
tar -zcv -f $backfile $backdir
```

```
vim /etc/crontab
45 2 * * * root sh /root/bin/backup.sh
```

5、以现今网络功能及维护来看，搭建一个"功能性強"的主机，还不如搭建一个"稳定且安全的主机"比较好一点

6、网站推荐

- <http://www.study-area.org>
- <http://linux.vbird.org>

1.3. 自我评估是否已经具备服务器搭建的能力

1、网管人员需要什么能力

- 是否具有 Linux 的基础概念

- 账号管理
- BASH
- 权限
- Process 与 Signal 的概念
- 简易的硬件与 Linux 相关性(如 mount)的认识, 日志文件的解析, daemon 的认识等
- 是否具备基础网络知识
 - IP
 - Netmask
 - route
 - DNS
 - daemon
 - port
 - TCP 数据包
- 是否能全身心投入
- 是否具有道德感与责任感
-

Chapter 2. 网络的基本概念

2. 1. 网络

1、不同的操作系统进行网络沟通，需要制定共同遵守的标准才行，这个标准由国际组织规范，你的系统里面只要提供可以加入该标准的程序代码，就能够通过这个标准与其他系统进行沟通

2. 1. 1. 什么是网络

1、网络就是几部计算机主机或者是网络打印机之类的接口设备，通过网线或者是无线网络技术，将这些主机与设备连接起来，使得数据可以通过网络介质(网线以及其他网卡等硬件)来传输的一种方式

2、发展

- 各自为政的"网络硬件与软件"技术发展：Ethernet & Token-Ring
- 以"软件"技术将硬件整合：ARPANET & TCP/IP
 - 可以在这些不同的网络硬件上面运行的软件技术，使得不同公司的计算机或数据可以通过这个软件来实现数据沟通
 - ARPANET 就是 TCP/IP 的雏形，TCP/IP 这种连接网络的技术被称之为 Internet
- 没有任何约束的因特网：Internet
 - Internet 就是使用 TCP/IP 的网络连接技术所连接起来的一个网络世界
 - Internet 仅是提供一个网络的连接接口
- 软硬件标准制定的成功带来的影响：IEEE 标准规范
 - 网卡：适配卡
 - 连接 Internet 则是需要去向网络服务提供商(Internet Service Provider, ISP)申请账号密码，凭账号密码才能连接
 - 网络硬件与软件非常多，最成功的却是以太网络(Ethernet)与 Internet，因为这两者都被"标准"所支持
 - 除了硬件之外，TCP/IP 这个 Internet 的通信协议也是有标准的，这些标准大部分都以 RFC(Request For Comments)形式发布的标准文件

2.1.2. 计算机网络组成组件

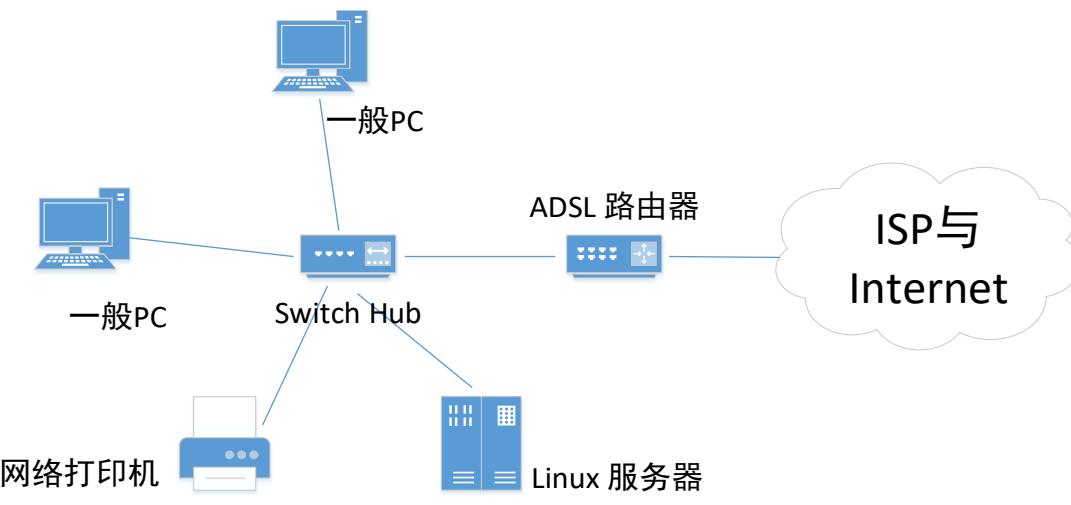


图 2-1 计算机网络连接示意图

1、节点(node):

- 节点主要是具有网络地址(IP)的设备的统称，图 2-1 中的一般 PC, Linux 服务器, ADSL 调制解调器与网络打印机都是网络中的节点，**但是集线器(Hub)不具有 IP，因此不是节点**

2、服务器主机(server):

- 就网络的连接方向来说，提供数据以"响应"给用户的主机，都可以被称为是一台服务器
- 例如 Yahoo 是 WWW 服务器，Linux 内核的 FTP(<ftp://ftp.kernel.org/pub/>)是一个文件服务器

3、工作站(workstation)或客户端(client):

- 任何可以在计算机网络输入的设备都可以是工作站
- 若以连接发起的方向来说，主动发起连接去"要求"数据的，就可以称为是客户端(client)

4、网卡(Network Interface Card,NIC):

- 内置或者是外接在主机上面的一个设备，主要用于提供网络连接，目前大都是使用具有 RJ-45 接口的以太网卡
- 一般节点上都具有一个以上的网卡，以实现网络连接功能

5、网络接口:

- 利用软件设计出来的网络接口，主要是提供网络地址(IP)的任务
- 一张网卡至少可以搭配一个以上的网络接口
- 每部主机内部也都拥有一个内部的网络接口，就是 loopback(lo)这个循环测试接口

6、网络形态或拓扑(topology):

- 各个节点在网络上面的链接方式，一般讲的是物理连接方式，例如图 2-1 的星形连接(star)的方式

7、网关(gateway):

- 具有两个以上的网络接口，可以连接两个以上不同的网段的设备
- 例如 **IP 路由器**就是一个常见的网关设备
- 而调制解调器不能算网关设备，因为调制解调器通常视为一个在主机内的网卡设备，我们可以在一般 PC 上通过拨号软件，将调制解调器仿真称为

一张实体网卡

2.1.3. 计算机网络的范围

1、由于各个节点的距离不同，连接的线缆与方式也有所差异，由于线缆的差异导致网络速度的不同，让网络应用方向也不一样。根据这些差异，早期习惯依据网络的大小范围将网络的种类定义如下几种

2、局域网络(Local Area Network, LAN)

- 节点之间的传输距离较近，可以使用较为昂贵的连接介质
- 可应用于科学运算的群集式系统、分布式系统、云端负载均衡系统等

3、广域网(Wide Area Network, WAN)

- 节点之间的传输距离较远，因此使用的连接介质的成本要低廉
- 网络应用大多为类似 E-mail, FTP, WWW 浏览等功能

4、除了 LAN 和 WAN 之外还有所谓的城域网网络(Metropolitan Area Network, MAN)，不过较少提及，只需要知道 LAN 与 WAN 即可

5、一般来说 LAN 是指较小的环境，如一栋大楼或一间学校。我们生活的周围有许多 LAN，这些 LAN 彼此连接在一起，全部的 LAN 连接在一起就是一个大型的 WAN 了

6、现在的环境跟以前不同了，光纤的速度可以达到 100Mbps/10Mbps 的下载/上传贷款了。另外，中国的教育网络都是连接在一起的，整个中国的教育网(CERNET)可视为一个局域网

2.1.4. 计算机网络协议：OSI 七层协议

1、整个网络的连接过程相当复杂，包括硬件、软件数据封装与应用程序的互相链接等，如果想要写一个将网络连接的全部功能都集中在一起的程序，那么当某个小环节出现问题时，整个程序都需要改写

2、我们将整个网络连接过程分成数个层次(layer)，每个层次都有特定独立的功能，每个层次的程序代码可以独立撰写，因为每个层次之间的功能并不会相互干扰，如此一来，当某个小环节出现问题时，只要将该层次的程序代码重新撰写即可

3、依据定义来说，越接近硬件的层次为底层(layer)，越接近应用程序的则是高层(layer)。不论是接收端还是发送端，每一层次只认识对方同一层次的数据

4、整个传送过程：通过应用程序将数据放入第七层的包裹，再将第七层的包裹放到第六层的包裹内，依序一直放到第一层的最大包裹内，然后传出去给接收端；接收端的主机就得由第一个包裹开始，依序将每个包裹拆开，然后一个一个交给对应负责的层次来查看，这就是 OSI 七层协议在层次定义方面需要注意的特色

5、包裹表面都有个重要的信息：来自哪里，要去哪里，接受者是谁，而包裹里面才是真正的数据。同样地，在七层协议中，每层都会有自己独特的头部数据，告知对方这里面的信息是什么，而真正的数据就附在后头



6、各层详细说明

➤ Layer 1 物理层(Physical Layer)

- 由于网络传输介质只能传输 0 与 1 这种比特位，因此物理层必须定义所使用的传输设备的电压与信号灯，同时还必须了解数据帧转成比特流的编码方式，最后连接实际传输介质并发送/接受比特信号

➤ Layer 2 数据链路层(Data-Link Layer)

- 这是比较特殊的一层，因为其下层是实体的定义，上层是软件封装的定义，因此第二层分为两个子层进行数据的转换操作
- 在偏硬件介质部分，主要负责的是 MAC(Media Access Control)，我们称这个数据包裹为 MAC 数据帧，MAC 是网络接口设备所能处理的主要数据包裹，这也是最终被物理层编码称比特流数据
- MAC 必须要经过通信协议来取得网络介质的使用权，目前最常使用的是 IEEE 802.3 的以太网络协议
- 偏软件的部分是由逻辑链接层(Logical Link Control, LLC)控制，主要在多任务处理来自上层的数据包数据(packet)并转成 MAC 的格式，负责的工作包括信息交换、流量控制、失误问题的处理等

➤ Layer 3 网络层(etworkLayer)

- IP(Internet Protocol)就是在这一层定义的
- 同时也定义出计算机之间的连接建立、终止与维持等
- 该层除了 IP 外，最重要的就是数据包能否到达目的地的路由概念了

➤ Layer 4 传输层(Transport Layer)

- 该层定义了发送端与接收端的连接技术(如 TCP、UDP 技术)，同时包括该技术的数据包格式，数据包的发送，流程的控制，传输过程的侦测检查与重新传送等，以确保各个资料数据包可以正确无误地到达目的端

➤ Layer 5 会话层(Session Layer)

- 该层定义了两个地址之间的连接信道的连接与中断
- 此外也可以建立应用程序之间的会话、提供其他加强型服务如网络管理、建立与断开、会话控制等
- 如果说传输层是在判断数据包是否可以正确到达目标，那么会话层则是在确定网络服务建立连接的确认

➤ **Layer 6 表示层(Presentation Layer)**

- 我们通过应用程序生成的数据格式不一定符合网络传输的标准编码格式，所以该层的主要操作就是：将来自本地端应用程序的数据格式转换(或者重新编码)成为网络的标准格式，然后再交给下面的传输层的协议来进行处理
- 在这个层次上面主要定义的是网络服务(或程序)之间的数据格式的转换，包括数据的加解密也在这个层次上处理

➤ **Layer 7 应用层(Application Layer)**

- 应用层本身不属于应用程序所有，而是在定义应用程序如何进入该层的沟通接口，以将数据接收或发送给应用程序，并最终展示给用户

7、事实上，OSI 七层协议只是一个参考的模型

- 目前的网络并没有什么很知名的操作系统在使用 OSI 七层协议的连接程序代码
- 不过 OSI 所定义出来的七层协议在解释网络传输的情况方面，可以解释的非常棒
- 大家都拿 OSI 七层协议来作为网络的教学与概念的理解
- 至于实际的联网程序代码，就交给 TCP/IP

2.1.5. 计算机网络协议：TCP/IP

1、OSI 七层协议的架构非常严谨，是学习网络的好材料，但是太过严谨，导致程序撰写相当不容易，造成了发展上的困扰

2、由 ARPANET 发展而来的 TCP/IP 也是使用 OSI 七层协议的观念，所以同样有分层的架构，只是将它简化为四层，在结构上面没有这么严谨，程序撰写会比较容易

3、连接过程详解：以 Yahoo 网站为例

- **应用程序节点**：打开浏览器，在浏览器的地址栏中输入网址，按下"Enter"键，此时网址信息与相关数据会被浏览器包成一个数据，并向下传给 TCP/IP 的应用层
- **应用层**：由应用层提供的 HTTP 通信协议，将来自浏览器的数据封装起来，并给予一个应用层报头，并向下传给传输层
- **传输层**：由于 HTTP 为可靠连接，因此将该数据丢入 TCP 封装内，并给予一个 TCP 封装的报头，并向下传给网络层
- **网络层**：将 TCP 数据封装到 IP 数据包内，再给予一个 IP 包头(主要就是来源与目标的 IP)，并向下传给网络接口层
- **网络接口层**：如果使用以太网络时，此时 IP 会依据 CSMA/CD 的标准，封装到 MAC 数据帧中，并给予 MAC 帧头，再转成比特流后，利用传输介质发送到远程主机上
- 等到 Yahoo 收到你的数据包，再反向拆开，交给对应层次进行分析，最后就让 Yahoo 的 WWW 服务器软件获知你想要的数据，该服务器软件再根据

你的要求取得正确资料后，又依据上述流程，一层层封装起来，最后传递到你手上

4、由于网络介质一次传输的数据量是有限的，因此如果要被传输的数据太大时，在各层的封装中，就需要将数据先拆开放到不同的数据包中，再给数据包一个序号，好让目的端的主机能够利用这些序号再重新将数据整合回来

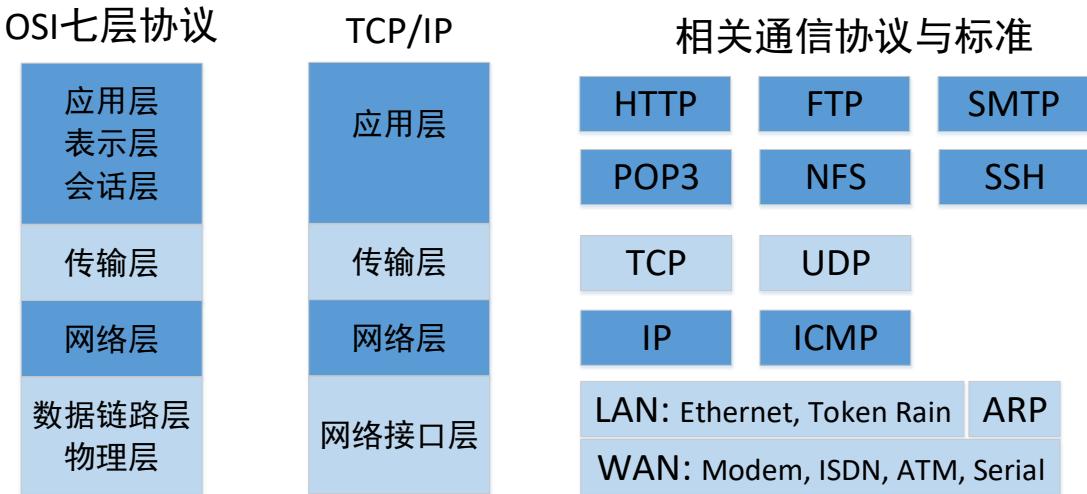


图 2-2 OSI 与 TCP/IP 协议的相关性

2. 2. TCP/IP 的网络接口层的相关协议

1、TCP/IP 最底层的网络接口层主要与硬件的关系比较密切

2. 2. 1. 广域网使用的设备

1、广域网使用的设备价格较为低廉，不过广域网使用到的设备非常多，一般用户通常会接触到的主要是 ADSL 调制解调器或者是光纤到接入，以及 Cable Modem 等

2、传统电话拨号连接：通过 PPP 协议

- 早期网络大概都只能通过调制解调器加上电话线以及计算机的九针串行接口(以前鼠标或游戏杆的插孔)，然后通过 Point-to-Point Protocol(PPP 协议)配合拨号程序来取得网络 IP 参数
- 当电话拨号连接建立后，就不能够使用电话了
- PPP 支持 TCP/IP、NetBEUI、IPX/SPX 等通信协议，所以使用度非常广

3、整合服务数字网络(Integrated Services Digital Network, ISDN)

- 利用现有的电话线路来实现网络连接的目的，只是连接的两端都需要有 ISDN 的调制解调器来提供连接功能。
- ISDN 的传输有多种信道可供使用，并且可以将多个信道整合应用，因此速度可以倍增长

4、非对称数字用户环路(Asymmetric Digital Subscriber Line, ADSL): 使用 PPPoE 协议

- 通过电话线来拨号后取得 IP 的一个方法，只不过这个方法使用的是电话的高频部分，与一般语音电话的频率不同，因此你可以一边使用 ADSL 上网，同时通过一个电话号码来打电话聊天

- 在中国，由于上传/下载的带宽不同，因此才称为非对称的环路
- ADSL 同样使用调制解调器，只是它通过的是 PPPoE(PPP over Ethernet)的方法，将 PPP 仿真在以太网卡上，因此你的主机只需要通过一张网卡来连接到调制解调器，并通过拨号程序来取得新的接口(ppp0)

2.2.2. 局域网使用的设备——以太网

1、在局域网的环境中，最常用的就是以太网，因为以太网已经标准化了，设备设置费用相对低廉，平时听到的网线或者是网络介质，几乎都是使用以太网来搭建的环境

2、整个网络世界并非只有以太网这个硬件接口

3、以太网的速度与标准

- 以太网的流行主要原因是它已经成为国际公认的标准
- 早先 IEEE 所制定的以太网络标准为 802.3 的 IEEE10BASE5
 - 10 代表传输速率为 10Mbps
 - BASE 代表采用基带信号来传输
 - 5 代表每个网络节点之间最长可达 500 米
- 由于网络传输信息就是 0 与 1，数据传输的单位为每秒多少 bit，也是 Mbits/second(Mbps)
- 早期的网线使用的是旧式的同轴电缆线，取而代之的是类似传统电话线的双绞线(Twisted Pair Ethernet)，IEEE 将这种线路的以太网络传输方法制定成为 10BASE-T 的标准
 - 10BASE-T 使用的是 10Mbps 全速运行且采用非屏蔽双绞线(UTP)的网线
 - 此外，10BASE-T 的 UTP 网线可以使用星形连接(star)，也就是以一个集线器为中心来连接各网络设备的一个方法
 - 通过星形连接，我们可以很简单地添加或移除设备，该项设计让以太网设备的销售额大幅提升
- 后来 IEEE 制定了 802.3u 这个支持到 100Mbps 传输速度的 100BASE-T 标准，该标准与 10BASE-T 差异不大，只是双绞线制作更精良，同时也支持了四对脚线的网线，也就是 8 蕊网线，这种网线成为五类 (Category5, CAT5) 的网线
- 传输速度增加，电磁效应相互干扰会增强，因此对网线的要求就更严格

表格 2-1 各种以太网的速度与网线等级

| 名称 | 速度 | 网线等级 |
|---------------------------|----------|--------------|
| 以太网(Ethernet) | 10Mbps | - |
| 快速以太网(Fast Ethernet) | 100Mbps | CAT5 |
| G 比特以太网(Gigabit Ethernet) | 1000Mbps | CAT 5e/CAT 6 |

4、以太网的网线接头(交叉/直连线)

- 网络的速度与线缆是有一定程度的相关性的，目前在以太网上最常见的接头就是 RJ-45 的网络接头，共有 8 蕊
- RJ-45 接头又因为每条蕊线的对应不同而分为 568A 和 568B 接头，这两款接头内的蕊线顺序对应如表格 2-2

表格 2-2 接头与蕊线的对应关系

| 接头名称/蕊线顺序 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|-----------|----|---|----|---|----|---|----|---|
| 568A | 白绿 | 绿 | 白橙 | 蓝 | 白蓝 | 橙 | 白棕 | 棕 |

| | | | | | | | | |
|------|----|---|----|---|----|---|----|---|
| 568B | 白橙 | 橙 | 白绿 | 蓝 | 白蓝 | 绿 | 白棕 | 棕 |
|------|----|---|----|---|----|---|----|---|

- 虽然目前的以太网线有 8 芯且两两成对，但实际使用的只有 1、2、3、6 芯，其他则是某些特殊用途的场合才会用到
- 主机与主机连接以及主机与集线器连接时，所使用的网线“线序”定义并不相同，因此由于接头不同网线又可分为两种：
 - 交叉线：一边为 568A 一边为 568B 的接头时称为交叉线，用在直接连接两台主机的网卡
 - 直连线：两边接头同为 568A 或 568B 时称为直连线，用在链接主机网卡与集线器之间的线缆

2. 2. 3. 以太网络传输协议：CSMA/CD

- 1、整个以太网的重心就是以太网卡，以太网的传输主要就是网卡对网卡之间的数据传递而已
- 2、每张以太网卡出厂时，就会赋予一个独一无二的卡号，那就是所谓的 MAC(Media Access Control)，理论上网卡卡号是不能修改的
- 3、IEEE802.3 的标准 CSMA/CD(Carrier Sense Multiple Access with Collision Detection)
 - 网络共享介质在单一时间点内，仅能被一台主机使用，否则就会冲突
 - **监听介质使用情况(Carrier Sense)**：A 主机要发送网络数据包前，需要先对网络介质进行监听，确认没有人在使用后，才能够发出数据帧
 - **多点传输(Multiple Access)**：A 主机发送的数据会被集线器复制一份，然后发送给所有连接到此集线器的主机。也就是说 A 发送数据，BCD 三部主机都能接收到，但由于目标是 D 主机，因此 B 与 C 会将数据帧丢弃，D 会抓下来处理
 - **冲突检测(Collision Detection)**：该数据帧附有检测能力，若其他主机例如 B 计算机刚好也在同时发送数据帧时，那么 A 与 B 送出的数据冲突，此时这些数据帧就被损毁，那么 A 与 B 就各自随机等待一个时间，然后重新发送
- 4、网络忙碌时，集线器信号灯闪个不停，但我的主机明明没有使用网络
 - 无论哪一台主机发送数据帧，所有的计算机都会接收到，因为集线器会复制一份该数据给所有计算机
- 5、我的计算机明明没有被入侵，为何我的数据会被隔壁计算机窃取
 - 我们只要在 B 计算机上安装一套监听软件，**这套软件将原本要丢弃的数据帧抓下来分析，并且加以重组，就能够知道原本 A 所送出的信息了，这也就是我们为什么建议重要数据再因特网上需要“加密”后再传输**
- 6、既然共享带宽设备只有一个主机可以使用，为何大家可以同时上网
 - 由于标准的数据帧在网卡与其他以太网媒体一次只能传输 1500bytes，因此大文件就需要拆成多个小数据包，然后一个个传送，每个数据包都要经过 CSMA/CD 机制
 - **并且即便只有一台主机在使用网络媒体，那么这部主机没法送一个数据包前也是需要等待一段时间的(96bit time)**
- 7、数据帧要多大比较好，能不能修改数据帧的长度
 - 超高速以太网络媒体若支持 Jumbo frame(巨型数据帧)，那么就能够将数据帧大小改为 9000bytes

2.2.4. MAC 的封装格式

| 前导码 8 Bytes | 目的地址 6 Bytes | 源地址 6 Bytes | 长度指示 2 Bytes | LLC数据 46-1500 Bytes | 帧校验序列 4 Bytes |
|----------------|-----------------|----------------|-----------------|------------------------|------------------|
|----------------|-----------------|----------------|-----------------|------------------------|------------------|

图 2-3 以太网的 MAC 数据帧

1、MAC 是整个网络硬件上面传送数据的最小单位

- 目的地地址与源地址就是网卡卡号(Hardware Address,硬件地址)
 - 每张网卡都有一个独一无二的卡号，卡号会在数据帧的帧头数据中使用
 - **硬件地址最小由 00:00:00:00:00:00 到 FF:FF:FF:FF:FF:FF(十六进制)**，这 6 bytes 中，前 3bytes 为厂商代码，后 3bytes 则是厂商自行定义的配置码
 - 在 Linux 系统中可以用<ifconfig>命令来查看网卡卡号
 - **特别注意，这个 MAC 的传送中，仅在局域网内生效，如果跨过不同的子网(A 与 C 通信要经过 B，分解为 A 与 B, B 与 C)，那么来源于目的的硬件地址就会跟着改变了，这是因为变成不同网卡之间的交流了，所以卡号当然不同**
 - 由于网卡卡号是跟着网卡走的，并不会因为重装操作系统而改变
- 为什么规定最小数据帧大小为 46bytes
 - 在 CSMA/CD 机制上面可算出要实现冲突检测，则数据帧总数最小需要 46bytes，扣除目的地址，源地址，长度指示，校验码(不算前导码)后，便得到最小的数据量为 46bytes
 - 如果传输的数据小鱼 46bytes，那么系统会主动填上一些填充码

2、**MAC 就是一直提到的数据帧，这个数据帧有两个很重要的数据，就是目标与来源的网卡卡号，因此我们又简称网卡卡号为 MAC 地址**

2.2.5. MTU(最大传输单位)

1、以太网数据帧所能传送的数据量最大为可达 1500bytes，这个数值就被称为 MTU(Maximum Transmission Unit,最大传输单元)

2、IP 数据包最大可达到 65535bytes

- IP 数据包是可以进行拆解的，然后才能放到 MAC 当中，等到数据都传到目的地，再由目的地主机将它组装回来就行了
- 如果 MTU 能够大的话，那么 IP 数据包拆解情况就会降低，数据包与数据包之间的等待时间(上面提到的 96bit time)也会减少，就能增加网络带宽的使用

3、为了达到**减少等待时间以及减少拆包次数**这个目的，所以才有 Gigabit 以太网对 Jumbo frame 的支持，这个 Jumbo frame 一般定义到 9000bytes，但是我们的 MTU 能改成 9000bytes 吗？

- 如果考虑到整个网络不建议这样做
- 我们的数据包要在 Internet 上传输，但是你无法确认所有网络设备都是支持那么大的 MTU，如果 9000bytes 通过一个不支持 Jumbo frame 的设备，好一点的情况是重组后传输，差的情况就直接丢弃了
- 因此 MTU 定义为 9000bytes 这种事情，大概仅能在内部网络环境中部署

2.2.6. 集线器、交换器与相关机制

1、**共不共享很重要**，集线器还是交换器

- 集线器(Hub)这个网络共享设备可能会发生冲突的情况，这是因为 CSMA/CD 的缘故
- 交换器(Switch)可以避免这种数据的冲突

2、交换器

- 交换器的等级非常多，仅讨论支持 ISO 第二层的交换器
- **交换器与集线器最大的差异在于交换器内有一个特别的内存，这个内存可以记录每个 Switch port 与其连接的 PC 的 MAC 地址**
- 当来自 Switch 两端的 PC 要互传数据时，每个数据帧将直接通过交换器的内存数据而传送到目标主机上，所以 Switch 不是共享设备，且 Switch 的每个端口(port)都有独立的带宽
- Switch 已经克服了数据包冲突的问题，因为有 Switch port 对应 MAC 的相关功能，所以 Switch 并不是共享设备
- 在选购 Switch 的时候，要选择支持全双工/半双工以及支持 Jumbo frame 的

3、带宽

- 10/100Mbps 的 Hub 连接 5 台主机，那么整个 10/100Mbps 是分给这五台主机的，所以 5 台主机总共只能用 10/100Mbps 而已
- 对于交换器，每个 port 都具有 10/100Mbps 的带宽，因此得看具体的传输行为了

4、什么是全双工/半双工(full-duplex/half-duplex)

- 8 芯网线实际上仅有两对被使用，一对用于发送，一对用于接受
- 如果两端的 PC 同时支持全双工时，那表示 Input/Output 均可达到 10/100Mbps，总带宽可达到 20/200Mbps
- 如果网络环境想要达到全双工时，使用共享带宽的 Hub 是不可能的，因为网线线序的关系，**无法使用共享带宽设备来达到全双工**
- 如果 Switch 也支持全双工模式，那么 Switch 两端的 PC 才能达到全双工

5、自动协调速度机制(auto-negotiation)

- 以太网卡是可以向下兼容的
- 新的 Hub/Switch 因为支持 auto-negotiation 功能呢，可以自动协调出最高的传输速度来通信

6、自动分辨网线的交叉或直连接口(Auto MDI/MDIX)

- Switch 若含有 Auto MDI/MDIX 功能时，会自动分辨网线的接口来调整连接

7、信号衰减造成的问题

- 电子信号是会衰减的，所以当网线过长导致电子信号衰减的情况严重时，就会导致连接质量不良，因此各个节点的网线长度是有限制的
- 一般来说，现今以太网络 CAT5 等级的网线大概都可以支持到 100 米的长度

8、结构化布线

- 所谓结构化布线是指将各个网络的组件分别拆开，分别安装与布置到企业内部，则未来想要升级网络硬件等级或者是移动某些网络设备时，只需要更改机柜的相关配线架，以及末端的墙上的预留孔与主机设备的连接就能够达到目的了

2.3. TCP/IP 的网络层相关数据包与数据

- 1、最常见的网络硬件接口为以太网，包括网线、网卡、Hub/Switch 等
- 2、以太网上面的传输使用网卡卡号为标准的 MAC 数据帧，配合 CSMA/CD 的标准来传送数据帧，这就是硬件部分
- 3、软件部分，Internet 其实就是 TCP/IP 这个通信协议的通称，Internet 是由 Inter NIC 所统一管理的，但其实它仅是负责分配 Internet 上面的 IP 以及提供相关的 TCP/IP 技术文件而已
- 4、Internet 最重要的就是 IP

2.3.1. IP 数据包的封装

- 1、目前因特网环境的 IP 有两个版本，一种是目前使用最广泛的 IPv4(Internet Protocol version 4，因特网协议第四版)，一种则是未来会热门的 IPv6
- 2、IPv4 记录的地址仅有 32 位，现在已经分配完毕
- 3、IPv6 地址可以达到 128 位，这样的 IP 数量几乎是用不完的
- 4、IP 数据包可达 65535bytes，在比 MAC 大的情况下，操作系统会对 IP 进行拆解的动作
- 5、IP 封装包头数据如图 2-4

| 4 bits | 4 bits | 8 bits | 3 bits | 13 bits | | |
|---------------------|----------|-----------------|----------------------|---------|--|--|
| Version | IHL | Type of Service | Total Length | | | |
| Identification | | Flags | Fragmentation Offset | | | |
| Time To Live | Protocol | | Header Checksum | | | |
| Source Address | | | | | | |
| Destination Address | | | | | | |
| Options | | | Padding | | | |
| Data | | | | | | |

图 2-4 IP 数据包的报头资料

- Version: 版本
- IHL(Internet Header Length): IP 报头长度
- Type of Service
 - 内容为 PPPDTRUU
 - PPP: 表示此 IP 数据包的优先级，目前很少使用
 - D: 若为 0 表示一般延迟，若为 1 表示低延迟
 - T: 若为 0 表示一般传输量，若为 1 表示高传输量
 - R: 若为 0 表示一般可靠度，若为 1 表示高可靠度
 - UU: 保留尚未使用
- Total Length(总长度): IP 数据包的总容量，包括报头与数据部分，最大可达 65535bytes
- Identification: 当 IP 数据包需要重组成较小的数据包，标志这些小数据包是否来自同一个 IP 数据包
- Flags(特殊标志)
 - 内容为"0DM"

- D: 若为 0 则表示可分段, 为 1 表示不可分段
- M: 若为 0 则表示此 IP 为最后分段, 若为 1 表示非最后分段
- Fragment Offset: 分段偏移
 - 表示这个 IP 分段在原始的 IP 数据包中的位置, 类似序号, 有了这个序号才能将所有小 IP 分段组合成原来的 IP 数据包
 - 通过 Total Length、Identification、Flags、Fragment Offset 就能够将小 IP 分段在接收端组合起来
- Time To Live(TTL, 生存时间): 表示这个 IP 数据包的生存时间, 范围为 0-255, 当这个 IP 数据包通过一个路由器时, TTL 就会减 1, 当 TTL 为 0 时, 这个数据包将会被直接丢弃(要让 IP 数据包通过 255 个路由器还是很难的)
- Protocol Number(协议代码)
 - 来自传输层与网络层本身的其他数据都是放置在 IP 数据包中, 我们可以在 IP 报头记载这个 IP 数据包内的数据是什么

| IP 内的代码 | 数据包协议名称 |
|---------|--|
| 1 | <ICMP>(Internet Control Message Protocol) |
| 2 | <IGMP>(Internet Group Management Protocol) |
| 3 | <GGP>(Gateway-to-Gateway Protocol) |
| 4 | <IP>(IP in IP encapsulation) |
| 6 | <TCP>(Transmission Control Protocol) |
| 8 | <EGP>(Exterior Gateway Protocol) |
| 17 | <UDP>(User Datagram Protocol) |

- Header Checksum(报头校验码): 用于检查 IP 报头是否存在错误
- Source Address(来源 IP 地址)
- Destination Address(目的 IP 地址)
- Options(其他参数): 额外功能, 包括安全处理机制, 路由记录, 时间戳、严格与宽松的来源路由等
- Padding(项目补齐): 若 Options 不足 32bits 时, 由 Padding 主动补齐

2.3.2. IP 地址的组成与分级

- 1、IP 其实是一种网络数据包, 而这个数据包的报头最重要的就是那个 32 位的来源与目标地址, 为了方便记忆, 我们也称这个 32bits 的数值为 IP 网络地址
- 2、IP 的组成是 32bits 的数值, 也就是由 32 个 0 与 1 组成的一连串数字, 那么当我们思考所有跟 IP 有关的参数时, 就应该要将参数想成是 32 位的数据
- 3、IP 最小可由 0.0.0.0 到 255.255.255.255, 在这一串数字中, 还可以分为 Net_ID(网络号码)与 Host_ID(主机号码)两部分

- 例如 192.168.0.0~192.168.0.255
- 前面三组数字 192.168.0 就是网络号码, 最后面一组数字则称为主机号码
- 同一个网络的定义是: 在同一个物理网段内, 主机 IP 具有相同的 Net_ID, 并且具有独特的 Host_ID
- 因此, 同一个 Net_ID 内不能具有相同的 Host_ID, 否则会产生冲突

- 4、IP 在同一网络的意义

- Net_ID 与 Host_ID 的限制
 - 在同一个网段内 Net_ID 是不变的, 而 Host_ID 则是不可重复的
 - 此外 Host_ID 在二进制的表示中, 不可同时为 0 也不可同时为 1, 全 0

表示整个网段的地址(Network IP), 而全 1 表示广播的地址(Broadcast IP)

- 对于 192.168.0 这个网段, 可用 IP 为 192.168.1-192.168.254
- 在局域网内通过 IP 广播传递数据
 - 在相同物理网段的主机如果使用相同的网络 IP 范围, 则这些主机都可以通过 CSMA/CD 的功能直接在局域网内用广播进行网络的连接
 - 也可以直接网卡对网卡传递数据(通过 MAC 数据帧)???
- 使用不同局域网在相同物理网段的情况
 - 在同一个物理网段内, 如果两部主机使用不同的 IP 网段地址, 则由于广播地址不同, 导致无法通过广播的方式来进行连接, 此时需要通过路由器来进行沟通才能将两个网络连在一起
- 网络的大小
 - 当 Host_ID 所占用的位越大, 级 Host_ID 数量越多, 表示同一个网络内可以设定主机的 IP 数量越多
- 单位公司内的计算集群, 或者是宿舍或者家里的计算机, 设置在同一个网络内是最方便的, 因为如此一来, 每一台计算机都可以直接通过 MAC 来进行数据的交流, 而不必由 Router(路由器)来进行数据包的传递了

5、IP 的分级

Class A: 0xxxxxxxx.xxxxxxxxxx.xxxxxxxxxx.xxxxxxxxxx ==>Net_ID 开头是 0
|---net---|-----host-----|

Class B: 10xxxxxxxx.xxxxxxxxxx.xxxxxxxxxx.xxxxxxxxxx ==>Net_ID 开头是 10
|-----net-----|-----host-----|

Class C: 110xxxxx.xxxxxxxxxx.xxxxxxxxxx.xxxxxxxxxx ==>Net_ID 开头是 110
|-----net-----|--host--|

Class D: 1110xxxx.xxxxxxxxxx.xxxxxxxxxx.xxxxxxxxxx ==>Net_ID 开头是 1110

Class E: 1111xxxx.xxxxxxxxxx.xxxxxxxxxx.xxxxxxxxxx ==>Net_ID 开头是 1111

Class A: 0.xx.xx.xx ~ 127.xx.xx.xx

Class B: 128.xx.xx.xx ~ 191.xx.xx.xx

Class C: 192.xx.xx.xx ~ 223.xx.xx.xx

Class D: 224.xx.xx.xx ~ 239.xx.xx.xx

Class E: 240.xx.xx.xx ~ 255.xx.xx.xx

- 因此, 只要知道 IP 第一个十进制数, 就能够大概了解到该 IP 属于哪个等级, 以及通网络的 IP 数量有多少
- **只需要记住 ABC 三个等级, D 是用来作为组播的特殊功能之用, 至于 E 是保留没有使用的网段**

2.3.3. IP 的种类与取得方式

1、在 IPv4 里面的两种 IP

- Public IP: 公共 IP, 经由 Inter NIC 所统一规划的 IP, 有这种 IP 才可以连上 Internet
- Private IP: 私有 IP 或保留 IP, 不能直接连上 Internet 的 IP, 主要用于局域网络内的主机连接规划

2、早在 IPv4 规划的时候就担心 IP 会有不足的情况, 而且为了应付某些企业内部的网络配置, 于是就有了私有 IP(Private IP)

- 私有 IP 也分别在 A、B、C 三个 Class 中各保留一段作为私有 IP 网段

- Class A: 10.0.0.0~10.255.255.255
- Class B: 172.16.0.0~172.31.255.255
- Class C: 192.168.0.0~192.168.255.255
- 这三段 Class 的 IP 是预留使用的，所以并不能直接作为 Internet 上面的连接使用
- **私有 IP 有几个限制**
 - 私有 IP 的路由信息不能对外传播(只能存在内部网络)
 - 私有 IP 作为来源或目的地址的数据包，不能通过 Internet 传送
 - 关于私有 IP 的参考记录(如 DNS)，只能限于内部网络使用
- 私有 IP 的好处
 - 由于它的私有路由不能对外直接提供信息，因此内部网络将不会直接被 Internet 上面的 Cracker 攻击，但是你也无法以私有 IP 来直接上网
 - 相当适合一些尚未具有 Public IP 的企业内部用来规划其网络的设置，否则当你随便指定一些可能是 Public IP 的网段来规划你企业内部的网络设置时，万一连上 Internet 就跟 Internet 原本的 Public IP 相同了
 - 在没有可用的公开网络时，想要和同学联网玩游戏，即在局域网内自己玩自己的连接游戏，只要规范好所有同学在同一私有 IP 网段中就可以了
 - 万一想要将这些私有 IP 送上 Internet：设定一个简单的防火墙加上 NAT(Network Address Transfer)服务，就可以通过 IP 伪装来使得私有 IP 的计算机也能连上 Internet

3、特殊的 Loopback IP 网段

- 还有一个奇怪的 Class A 的网络，即 lo 网络，lo 网络时当初被用来作为测试操作系统内部循环所用的一个网络，同时也能够提供给系统内部原本就需要使用网络接口的服务(daemon)所使用
- 当你没有在机器上安装网卡，但你又希望可以测试一下在你机器上面设置的服务器环境到底可不可以顺利工作，这是就可以利用这个所谓的内部回环网络了
- 这个网段在 127.0.0.0/8 这个 Class A，而且默认的主机(localhost)的 IP 是 127.0.0.1
- 因此当你启动了 WWW 服务器，然后在你主机的 X-Windows 上面执行 <http://localhost> 就可以直接看到主页了，不必安装网卡，测试很方便

4、IP 的取得方式

- 基本上，主机的 IP 与相关网络的设置方式主要有以下几种
- 直接手动配置(static):
 - 直接编辑配置文件(或使用特定软件)来设置你的网络
 - 常见于校园网络的环境中，以及向 ISP 申请固定 IP 的连接环境
- 通过拨号取得
 - 向你的 ISP 申请注册，取得账号密码后，你的 ISP 会通过他们自己的设置，让你的操作系统取得正确的网络参数
 - 此时你并不需要手动去编辑与配置相关的网络参数了
- 自动取得网络参数(DHCP)
 - 在局域网络内会有一台主机负责管理所有计算机的网络参数，你的网络启动时就会主动向该服务器要求 IP 参数，取得网络相关参数后，你的主机就能够自行设定好所有服务器给你的网络参数了

- 最常适用于企业内部，IP 路由器后端、校园网络与宿舍环境

2. 3. 4. Netmask、子网与 CIDR(Classless Interdomain Routing)

- 1、IP 是有等级的，而配置在一般计算机系统上面的则是 Class A、B、C。
- 2、如果定义一个局域网，使用的是 Class A，同一个 Class A 的网段内最多有 $256 \times 256 \times 254 = 16777214$ 台主机，如果利用 CSMA/CD 机制，那么网络会特别的拥堵，因为你需要接到一千多万台计算机对你的广播
- 3、可以再进一步细分 IP

- 对于 Class C 的网络号码占了 24 位，我们可以让第一个 Host_ID 被拿来作为 Net_ID，所以整个 Net_ID 就有 25bit，至于 Host_ID 就减少为 7bits
- 在这样的情况下，原来一个 Class C 的网络就被划分为两个子网，每个子网就有 $256/2=126$ 个可用 IP 了

4、Netmask

- Netmask 用于实现子网的划分(Subnet mask，即子网掩码)
- 这个 Netmask 是定义网络最重要的一个参数，也最难理解
- 以 192.168.0.0~192.168.0.255 这个网络为例
 - 这个 IP 网段可以分为 Net_ID 与 Host_ID
 - 既然 Net_ID 是不可变的，那就假设它所占据的位已经被用完了(全为 1)
 - 而 Host_ID 是可变的，就将它想成是保留的(全为 0)

第一个 IP: 11000000.10101000.00000000.00000000
 最后个 IP: 11000000.10101000.00000000.11111111
 |-----Net_ID-----| -Host_ID- |
 Netmask: **1111111.1111111.1111111.00000000**
 : 255 . 255 . 255 . 0
- Class A B C 三个等级的 Netmask 可表示为
 - Class A: 11111111.00000000.00000000.00000000 ==> 255.0.0.0
 - Class B: 11111111.11111111.00000000.00000000 ==> 255.255.0.0
 - Class C: 11111111.11111111.11111111.00000000 ==> 255.255.255.0
- 因此在 192.168.0.0~192.168.0.255 这个 IP 网段里面相关的网络参数为
 - Netmask: 255.255.255.0 <== 网络定义中，最重要的参数
 - Network: 192.168.0.0 <== 第一个 IP
 - Broadcast: 192.168.0.255 <== 最后一个 IP
 - 可用以设定成主机的 IP: 192.168.0.1~192.168.0.254

5、子网划分

- 仍然以 192.168.0.0~192.168.0.255 为例
- 原本的 Class C 的 Net_ID 与 Host_ID

| | |
|-------------------------------------|-------------------------|
| 11000000.10101000.00000000.00000000 | Network:192.168.0.0 |
| 11000000.10101000.00000000.11111111 | Broadcast:192.168.0.255 |
| -----Net_ID----- -Host_ID- | |
- 换分成两个子网后的 Net_ID 与 Host_ID(将原本 Host_ID 的第一位作为 Net_ID)

| | |
|--|--|
| 11000000.10101000.0000000 0 0000000 | |
| 11000000.10101000.0000000 1 0000000 | |
| -----Net_ID----- -Host_ID-- | |

- 第一个子网

| | | |
|--------------------------------------|------------|-------------------------|
| 11000000.10101000.00000000. 0 | 0000000 | Network:192.168.0.0 |
| 11000000.10101000.00000000. 0 | 1111111 | Broadcast:192.168.0.127 |
| -----Net_ID----- | -Host_ID-- | |
| 11111111.11111111.11111111. 1 | 0000000 | Netword:255.255.255.128 |
- 第二个子网

| | | |
|--------------------------------------|------------|-------------------------|
| 11000000.10101000.00000000. 1 | 0000000 | Network:192.168.0.128 |
| 11000000.10101000.00000000. 1 | 1111111 | Broadcast:192.168.0.255 |
| -----Net_ID----- | -Host_ID-- | |
| 11111111.11111111.11111111. 1 | 0000000 | Netword:255.255.255.128 |

- 如有需要，还能继续划分子网

6、无类别域间路由 CIDR(Classless Interdomain Routing)

- 一般来说，如果知道了 Network 以及 Netmask 之后，就可以定义出该网络的所有 IP 了，因为 Netmask 就可以推算出来 Broadcast 的 IP
- 因此，我们常常会以 Network 以及 Netmask 来表示一个网络
- 事实上，由于网络细分的情况太严重，为了担心路由信息过于庞大导致网络效能不佳，在某些特殊情况下，反而将 Net_ID 借用来作为 Host_ID，这样就能够将多个网络写成一个了
- 例如 192.168.0.0/16，反而将 192 开头的 Class C 变成了 Class B，这种打破原本 IP 代表等级的方式(通过 Netmask 规范)就被称为无类别域间路由 (CIDR)
- 其实无须理会无类别域间路由，其实 Network/Netmask 的写法，通常就是 CIDR 的写法

2.3.5. 路由概念

- 1、不同网段的主机是不能传递数据的
- 2、主机想要发送数据时，它主要参考的是“**路由表**”，每台主机都有自己的“路由表”

3、数据传输流程

- 查询 IP 数据包的目标 IP 地址
- 查询是否位于本机所在的网络路由表中
 - 当发现目标 IP 与本机 IP 的 Net_ID 相同时，则通过局域网之间传输消息
- 查询默认路由(Default Gateway)
 - 首先分析路由表中是否有其他相符合的路由设置值，如果没有的话，就直接将该 IP 数据包送到默认路由器去
- 送出数据包至 Default Gateway 后，不理会数据包流向
 - 默认路由接收到数据包后，会根据上述流程分析自己的路由信息，然后继续传输，直到到达目的主机

- 4、Gateway/Router：网关/路由器的功能就是在负责不同网络之间的数据包传递 (IP Forwarding)

- 由于路由器具有 IP Forwarding 的功能，并且具有管理路由的能力，所以可以将来自不同网络之间的数据包进行传递

- 此外，你的主机与你主机配置的 Gateway 一定要在同一个网段内

- 5、事实上，Internet 上的路由协议与变化是相当复杂的，因为 Internet 上面的路

由并不是静态的，它可以随时因为环境的变化而修订每个数据包的传送方向

2.3.6. 观察主机路由: Route

1、路由一旦设置错误，将会造成某些数据包完全正确的传出去

2、<route>

- route [-n]
- -n: 将主机名以 IP 的方式显示
- 输出字段含义解释
 - Destination: Network 的意思
 - Gateway: 该接口的 Gateway 的 IP, 若为 0.0.0.0, 表示不需要额外的 IP
 - Genmask: 就是 Netmask, 与 Destination 组合成为一台主机或网络
 - Flags: 共有多个标志来表示该网络或主机代表的意义
 - U: 代表该路由可用
 - G: 代表该网络需要经由 Gateway 来帮忙传递
 - H: 代表该行路由为一台主机, 而非一整个网络
 - Iface: 就是 Interface(接口)的意思

2.3.7. IP 与 MAC: 网络接口层的 ARP 与 RARP 协议

1、Internet 上面最重要的就是 IP 了，也会计算所谓的局域网与路由，但是事实上用于传递数据的分明就是以太网，以太网主要是用网卡卡号(MAC)。

2、ARP(Address Resolution Protocol, 网络地址解析)协议，以及 RARP(Reverse ARP, 反向网络地址解析)协议

3、想要了解某个 IP 配置于哪张以太网卡上时，我们主机会对整个局域网发送出 ARP 数据包，对方收到 ARP 数据包后就会返回它的 MAC 给我们，我们的主机就会知道对方所在的网卡，接下来就能开始传递数据了

4、如果每次传递都需要重新来一遍这个 ARP 协议是很麻烦的，因此，当使用 ARP 协议取得目标 IP 与它的网卡卡号后，就会将该笔记录写入主机的 ARP table，记录 20 分钟

5、<arp>

- 取得/修改本机 ARP 表格内的 IP/MAC 对应数据
- arp -[nd] hostname
- arp -s [hostname] [Hardware_address]
- -n: 将主机名以 IP 的形态显示
- -d: 将 hostname 的 hardware_address 由 ARP table 中删除掉
- -s: 设定某个 IP 或 hostname 的 MAC 到 ARP table 中

2.3.8. ICMP 协议

1、ICMP 的全称是 Internet Control Message Protocol，即因特网信息控制协议

2、基本上，ICMP 是一个错误检测与报告的机制，最大的功能就是可以确保我们网络的连接状态与连接的正确性

3、ICMP 也是网络层的重要数据包之一，这个数据包并非独立存在，而是纳入到 IP 数据包中，也就是说，ICMP 同样是通过 IP 数据包来进行数据传送的

4、如何利用 ICMP 来检验网络的状态？最简单的指令就是 ping 与 traceroute

2.4. TCP/IP 的传输层相关数据包与数据

- 1、网络层的 IP 数据包只负责将数据送到正确的目标主机去，但这个数据包到底会不会被接受，或者是有没有被正确接受，就不是 IP 的任务了，那就是传输层的任务了
- 2、从图 2-2 可以看出传输层有两个重点，一个是面向连接的 TCP 数据包，一个是无连接的 UDP 数据包，这两个数据包很重要，数据能不能正确被送达目的地，与这两个数据包有很大关系

2.4.1. 面向连接的可靠的 TCP 协议

- 1、网络层 IP 之上是传输层，而传输层的数据打包成什么？最常见的就是 TCP 数据包，这个 TCP 数据包必须要能够放到 IP 数据“袋”中才行

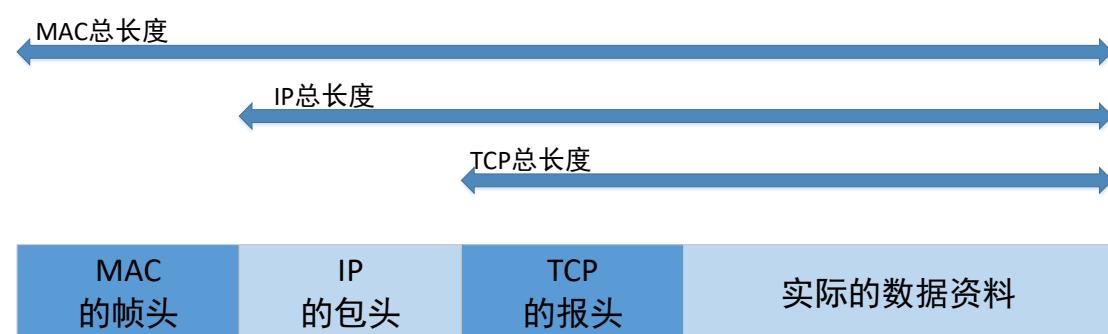


图 2-5 各数据包之间的相关性

| 4 bits | 6 bits | 6 bits | 8 bits | 8 bits | | |
|--------------------|----------|--------|------------------|--------|--|--|
| Source Port | | | Destination Port | | | |
| Sequence Number | | | | | | |
| Acknowledge Number | | | | | | |
| Data offset | Reserved | Code | Window | | | |
| Checksum | | | Urgent Pointer | | | |
| Options | | | Padding | | | |
| Data | | | | | | |

图 2-6 TCP 数据包的报头信息

2、TCP 数据包报头内容解析

- **Source Port&Destination Port(源端口&目标端口)**
 - 什么是端口？IP 数据包的传送主要是利用 IP 地址链接两端，这个连接的通道连接到 port 上面
 - 这个目标与来源 port 的记录，可以说是 TCP 数据包上最重要的参数了
- **Sequence Number(数据包序号)**
 - 由于 TCP 数据包必须要带入 IP 数据包当中，所以如果 TCP 数据太大时，就需要进行分段，这个 Sequence Number 就是记录每个数据包的序号，

可以让接收端重新将 TCP 的数据组合起来

- 有点类似于当 IP 数据包大于 MAC 数据帧时，需要将 IP 数据包重组为较小的数个 IP 数据包
- **Acknowledge Number(回应序号)**
 - 为了确认主机端确实收到 Client 端发出的数据包，我们 Client 端当然希望能够收到主机方面的响应，就是这个 Acknowledge Number 的用途了
- **Data offset(数据补偿)**
 - 由于 Options 字段的长度是非固定的
- **Reversed(保留): 未使用的保留字段**
- **Code(Control Flag,控制标识码)**
 - 当我们在进行连接的时候，必须要说明这个连接的状态，好让接收端了解这个数据包的主要作用
 - 这个字段共有 6 bits，分别代表 6 个句柄，若为 1 则为启动
 - **URG(Urgent):** 若为 1 则代表该数据包为紧急数据包，接收端应该要紧急处理，且图 2-15 当中的 Urgent Pointer 字段也会被启动
 - **ACK(Acknowledge):** 若为 1 代表这个数据包为响应数据包，则与上面提到的 Acknowledge Number 有关
 - **PSH(Push function):** 若为 1 时，代表要求对方立即传送缓冲区内的其他对应数据包，而无需等待缓冲区满了才传送
 - **RST(Reset):** 如果 RST 为 1 的时候，表示连接会被马上结束，而无需等待终止确认手续，也就是说，这是个强制结束的连接，且发送端已断线
 - **SYN(Synchronous):** 若为 1，表示发送端希望双方建立同步处理，也就是要求建立连接，带有 SYN 标志的数据包表示“主动”要连接到对方的意思
 - **FIN(Finish):** 若为 1，表示传送结束，所以通知对方数据传输完毕，是否同意断线，只是发送者还在等待对方的响应而已
- **Window(滑动窗口)**
 - 主要是用来控制数据包流量的，可以告知对方目前本机的缓冲器(Receive Buffer)还可以接受数据包
 - 当 Window=0 时，代表缓冲器已经满额，所以应该要暂停传输数据
 - Window 的单位是 byte
- **Checksum(确认校验码)**
 - 当数据要由发送端送出前，会进行一个校验的动作，并将该动作的校验值标注在这个字段上
 - 接受者接收到这个数据包之后，会再次对数据包进行验证，并且对比原发送的 Checksum 值是否相符，如果相符就接受，不符就损毁，要求对方重新发送
- **Urgent Pointer(紧急数据)**
 - 这个字段是在 Code 字段内的 URG=1 时才会产生作用，可以告知紧急数据所在的位置
- **Options(任意数据)**
 - 目前仅用于表示接收端可以接受的最大数据段容量
- **Padding(补足字段)**
 - 由于 Options 字段长度不定，补足由 Padding 来补齐至 32 bits

3、通信端口

- TCP 报头数据中，最重要的就是 16 位的两个量，也就是来源与目标的端口，由于是 16 位，因此目标与来源端口最大可达 65535 号
- 端口是用于达成连接通道，可以将 IP 理解为门牌，而端口则为这个门牌所对应的建筑的楼层，所选的端口(楼层)是否有人在服务呢？这就取决于有没有程序在该端口提供服务了
- IP 是门牌，端口是楼层，真正提供服务的是该楼层的那个人(程序)
- 一台主机上面有很多服务，我们跟这台主机进行连接时，该主机怎么知道我们要的数据时 WWW 还是 FTP 呢？，就是通过端口

4、特权端口(Privileged Ports)

- 既然网络是双向的，一定有一个发起端，问题是，到底要连接到服务器取得什么呢？
- 也就是说，哪段程序应该在哪个端口执行，以让大家知道该端口就是提供那个服务的，如此一来，才不会造成广大用户的困扰
- 因此 Internet 上面有很多规范好的固定 port(well-known port)，这些 port number 通常小于 1024，且是提供给许多知名的网络服务软件用的
- 在 Linux 环境下，各网络服务与 port number 的对应默认给它卸载 /etc/services 文件内
- 常见的端口与网络服务对应见表格 2-3

表格 2-3 常见的端口与网络服务的对应

| 端口 | 服务名称与内容 |
|-----|------------------------------------|
| 20 | FTP-data，文件传输协议所使用的主动数据传输端口 |
| 21 | FTP，文件传输协议的命令端口 |
| 22 | SSH，较为安全的远程连接服务 |
| 23 | Telnet，早期的远程连接服务器软件 |
| 25 | SMTP，简单邮件传递协议，用在作为 Mail Server 的端口 |
| 53 | DNS，用在作为名称解析的域名服务器 |
| 80 | WWW，就是全球信息网服务器 |
| 110 | POP3，邮件接受协议，办公使用的收信软件都是通过它 |
| 443 | HTTPS，有安全加密机制的 WWW 服务器 |

- 小于 1024 以下的端口要启动时，启动者的身份必须要是 root 才行，所以才叫做特权端口，不过如果是 Client 端的话，由于 Client 端都是主动向 Server 端要数据，所以 Client 端的 port number 就是用随机取一个大于 1024 且没有在用的 port number 即可

5、Socket Pair

- 由于网络是双向的，要达成连接的话需要服务器与客户端均提供了 IP 与端口才行，因此，我们常常将这个成对的数据称之为 Socket Pair
- 来源 IP+来源端口(Source Address+Source Port)
- 目的 IP+目的端口(Destination Address+Destination Port)
- 由于 IP 与端口常常连在一起说明，因此网络寻址常常使用"IP:port"来说明

2.4.2. TCP 的三次握手

1、TCP 被称为可靠的数据传输协议，主要是通过许多机制来实现的，其中最重要的就是三次握手的功能，在建立连接之前必须通过三个确认动作

➤ **数据包发起**

- 当客户端想要对服务器端连接时，就必须要送出一个要求连接的数据包，此时客户端必须随机选取一个大于 1024 的端口来作为程序沟通的接口，
- 然后在 TCP 报头中，必须要带有 SYN(synchronous)的主动连接(SYN)，并且记下发送出连接数据包给服务器端的序号(Sequence number=10001)

➤ **数据包接收与确认数据包传送**

- 当服务器接到这个数据包，并且确定要接收这个数据包后，就会开始制作一个同时带有 SYN=1， ACK(Acknowledgement)=1 的数据包，其中那个 Acknowledge 的号码是要给 Client 端确认用的，所以该数字会比 A 步骤里面的 Sequence 号码多一号(ack=10001+1=10002)
- 服务器必须要确认客户端可以接收到服务端的数据包才行，因此也会发送一个 Sequence number=20001，并且开始等待客户端给我们服务器端的回应

➤ **回送确认数据包**

- 当客户端收到来自服务器端的 ACK 数字(10002)后，就能够确认之前那个要求数据包被正确接受了，接下来如果客户端也同意与服务器端建立连接时，就会再次发送一个确认数据包(ACK=1)给服务器，ack=20001+1=20002

➤ **取得最后确认**

- 若一切都顺利，在服务器端收到带有 ACK=1 且 ack=20002 序号的数据包后，就能够建立这次连接了

2.4.3. 无连接的 UDP 协议

1、UDP 全称是 User Datagram Protocol，即用户数据报协议

2、与 TCP 不同，UDP 不提供可靠的传输模式，因为它不是面向连接的机制，这是因为在 UDP 的传送过程中，接收端在接收到数据包后，不会回复响应数据包(ACK)给发送端，所以数据包并没有像 TCP 数据包有较为严密的检查机制

| | |
|----------------|------------------|
| 16 bits | 16 bits |
| Source Port | Destination Port |
| Message Length | Checksum |
| Data | |

图 2-7 UDP 数据包的报头资料

3、TCP 数据包确实比较可靠，因为通过三次握手，不过也由于三次握手的缘故，TCP 数据包的传输速度会较慢

4、UDP 数据包由于不需要确认对方是否正确接收到数据，故报头数据较少，所以 UDP 就可以在 Data 处填入更多的数据了，UDP 适合需要实时反映的一些数据流

5、很多软件同时提供 TCP\UDP 的传输协议

- 查询主机名的 DNS 服务就提供了 UDP/TCP 协议
- 由于 UDP 较为快速，Client 先用 UDP 来与服务器连接，但是当使用 UDP 连接却还是无法取得正确数据时，便转为较为可靠的 TCP 传输协议了
- 这样可以兼顾快速与可靠的传输

2.4.4. 网络防火墙与 OSI 七层协议

1、数据的传送其实就是数据包的发出与接受的动作，并且不同数据包上面都有不一样的头部(header)信息

2、数据包上面通常会具有四个基本信息

- 来源与目的 IP
- 来源与目的 Port number

3、数据包过滤式的防火墙可以阻挡掉一些可能有问题的数据包

- 数据包的头部包含了很多的信息，我们可以利用一些防火墙机制与软件来进行数据包报头的分析，并设定分析的规则
- 若发现某些特定 IP，特定的端口或者是特定的数据包信息(SYN/ACK 等)，那么就将该数据包丢弃，这就是最基本的防火墙原理

4、以 OSI 七层协议来说，每一层可以阻挡的数据有：

- 第二层：可以针对来源与目标的 MAC 进行阻挡
- 第三层：主要针对来源与目标 IP，以及 ICMP 的类别进行阻挡
- 第四层：针对 TCP/UDP 的端口进行阻挡，也可以针对 TCP 的状态(code)来处理

2.5. 连上 Internet 前的准备事项

1、Internet 上面使用的是 TCP/IP 这个通信协议，所以我们就需要 Public IP 来连接上 Internet

2、问题：为什么我不知道 Yahoo 的主机 IP，但是我的主机却可以连接到 Yahoo 主机上

2.5.1. IP 地址、主机名与 DNS 系统

1、要连上 Internet 就需要有 TCP/IP 才行，尤其是最重要的 IP，不过人类对于 IP 这一类的数字并不具有敏感性，即使 IP 已经简化为十进制了

- 计算机有主机名称，我们就将主机名称与它的 IP 对应起来，那么要连上该计算机时
- 只要知道该计算机的主机名称就好了，因为 IP 已经对应到名称了

2、主机名称(Host Name)对应 IP 的系统，就是著名的 Domain Name System(DNS)

- DNS 这个服务最大功能就是在进行“主机名称与该主机 IP 的解析”这一项协议
- 当我们在浏览器上输入 <http://tw.yahoo.com> 时，计算机首先就会通过 DNS 主机查询 <http://tw.yahoo.com> 的 IP 后，再将查询到的 IP 结果回应给我的

- 浏览器，于是浏览器就可以利用该 IP 连上主机了
 - 由于计算机需要向 DNS 服务器查询 Host Name 对应 IP 的信息，那么那部 DNS 主机的 IP 就必须要在我的计算机里面设置好才行，并且需要时输入 IP，不然计算机怎么连接到 DNS 服务器去要求数据呢，在 Linux 中，DNS 主机 IP 设定就在/etc/resolv.conf 这个文件里
- 3、目前各大 ISP 都会提供他们的 DNS 服务器的 IP 给他们的用户，好设定客户在自己计算机的 DNS 查询主机
- 如果你忘了或者是你使用的环境中并没有提供 DNS 主机，那么就设定 Hinet 那个最大的 DNS 服务器，IP 是 168.95.1.1
 - 设定好 DNS 之后才能使用主机名，否则就只能用 IP 才能上网

Chapter 3. 局域网架构简介

3.1. 局域网的连接

3.1.1. 局域网的布线规划

1、大部分狭义的定义中，将局域网定位在一个采用星形拓扑连接的实体网络中，再通过 IP 网段来连接在一起的情况，因此这个网络是怎么连接在一起的，以及 IP 网段如何规划的，就显得非常重要

2、**网络布线是"数十年大计"中最重要的环节**

- 服务器主机能力不够时换主机就好了，Switch 交换力不足时换 Switch 就好了，但是如果布线不良，难道要拆掉房子将管线挖出来重新安装设置吗

3、在单纯的环境中，我们可以利用一个 Switch 为中心来连接所有设备的星形拓扑结构来设计局域网，需要考虑的只是我的 Linux 服务器放置的地方

3.1.1.1. Linux 直接联网——让 Linux 与一般 PC 地位相同

1、最简单的连接方式如图 3-1

- 在这种连接模式中，Linux 与一般 PC 或打印机都是同等地位
- **若不着急连上 Internet，每个设备都给予一个相同网络的私有 IP 就可以进行网络连接工作了**
- 当需要连上 Internet 时，每个计算机都可以自己直接通过拨号程序连接到 Internet
 - 由于拨号时在每台机器上面"额外增加一个实体的 ppp0 接口"，此时你的系统内就会有两个可以使用的 IP 了(public 和 private)
 - 因此，拨号上网之后每部主机还是可以使用原有的局域网内的各项服务，而无需改变原本设置妥当的私有 IP
- 这样的环境对于企业来说不好管理，因为无法掌握每个员工实际上网的情况，没法对员工进行任何网络控制，并且由于网络内外部(LAN 与外部环境)没有明确的界限，网管人员对于进入客户端的数据包是没有任何管理能力的，因此对于企业，不建议采用这样的环境

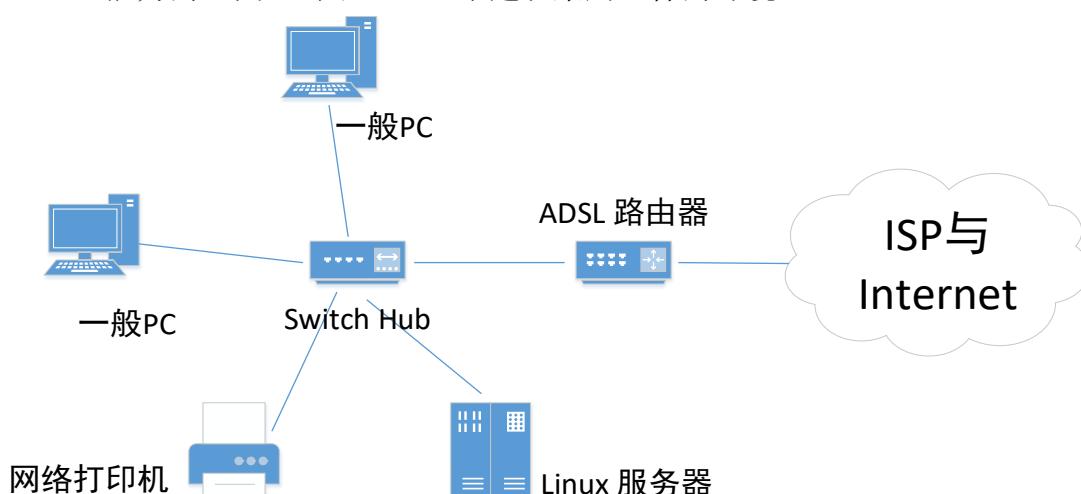


图 3-1 Linux 服务器取得 Public IP 的连接方式之一

3.1.1.2. Linux 直接联网——让 Linux 与一般 PC 处于不同的地位

1、如果有多个可用的 Public IP，并且你的 Linux 服务器主要提供 Internet 的 WWW 或 Mail 服务，而不是作为内部的文件服务器使用，那么将 Linux 服务器与内部网络分开也是个可行的方法，如图 3-2

- 所有 LAN 内的计算机与相关设备都会在同一个网络内所以在 LAN 内传输速度没有问题
- 此外这些计算机要连接到 Internet，必须要通过 IP 路由器，所以你也可以在 IP 路由器上面定义简单的防火墙规则
- 如果 IP 路由器可以更换为功能全面的路由设备时，就可以在该设备上定义较为完整的防火墙规则，完善对内部主机的管理，并且方便维护

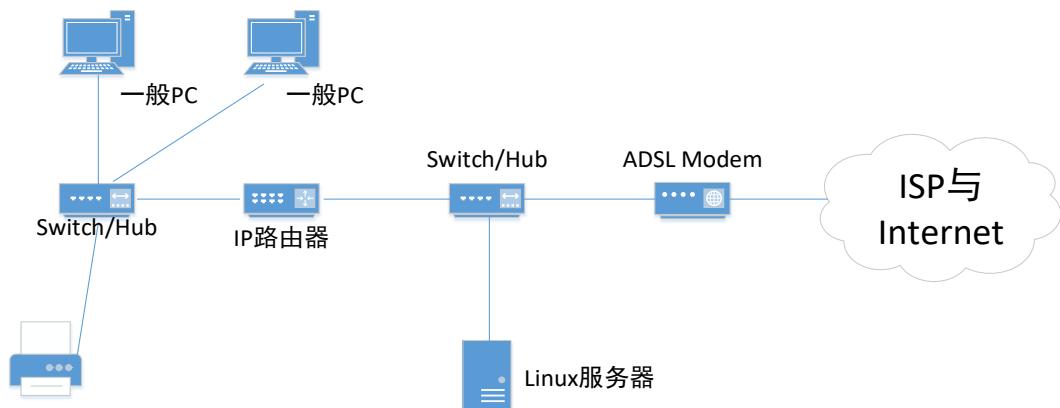


图 3-2 Linux 服务器取得 Public IP 的连接方式之二(具有多个可用 IP 情况)

3.1.1.3. Linux 直接联网——让 Linux 直接管理 LAN

1、如果不购买 IP 路由器，可以直接利用 Linux 服务器来管理，如图 3-3

- 这种情况下，不论你有多少个 IP 都是适用的，尤其是当你只有一个 Public IP 时，就非得使用这种方式不可
- **让 Linux 作为 IP 路由器的功能非常简单，同时 Linux 必须具备两张网卡，分别对外对内**
- 由于 Linux 服务器可以作为内部网络对外的防火墙之用，由于 Linux 防火墙的效率比较高加上配置也很简单，所以功能不错
- 鸟哥推荐这种
- 不过，服务器提供的网络服务越简单越好，因为这样一来主机的资源可以完全被某个程序所用，不会互相影响，而且当主机被攻击时，也能够立即了解是哪个环节出了问题
 - 对于图 3-3 这种情况
 - 由于内部 LAN 是需要通过 Linux 才能连接到 Internet，所以当 Linux 宕机时，整个对外的连接就中断了
 - 此外，这也会造成 Linux 的服务结构的复杂化，进而会造成维护上的困难
 - 对于小型局域网来说，这种架构还是有一定应用价值的

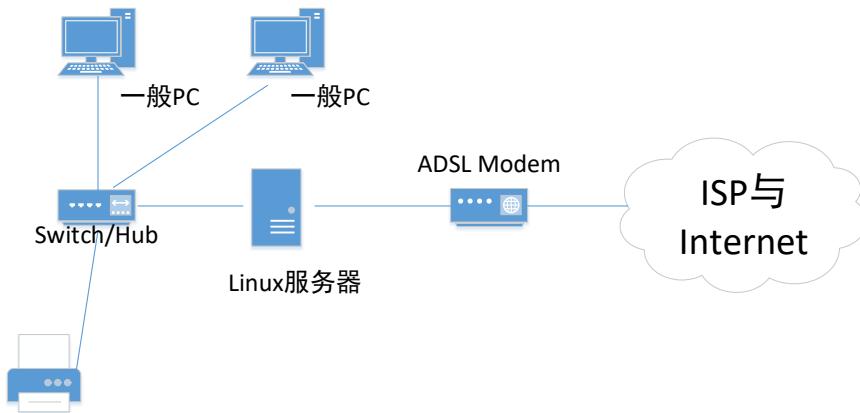


图 3-3 让 Linux 管理 LAN 的布线情况

3.1.1.4. Linux 放在防火墙后——让 Linux 使用 Private IP

1、比较大型的企业通常会将他们的服务器主机放置在机房内

- 主要在 LAN 的环境下，再通过防火墙的代理功能，将来自 Internet 的数据包先经过防火墙过滤后才进入到服务器
- 如此一来，可在防火墙端就过滤掉一些莫名其妙的探测与攻击，当然会比较安全

2、如果可以的话，将 IP 路由器换成 Linux 主机来架设防火墙，也是一个不错的选择

- 现在计算机天天升级，升级后的旧配设备其实就可以作为 Linux 防火墙之用
- **防火墙不需要硬盘与高清显示器或者 CPU，只要有不错的网络接口就能达到不错的防火墙性能**

3、Linux 服务器若放在 LAN 里面(使用 Private IP)，则当你要对 Internet 提供网络服务时，防火墙的规则将变得相当复杂，因为需要进行数据包传递的任务，在某些比较麻烦的协议中，可能会造成配置方面的困扰

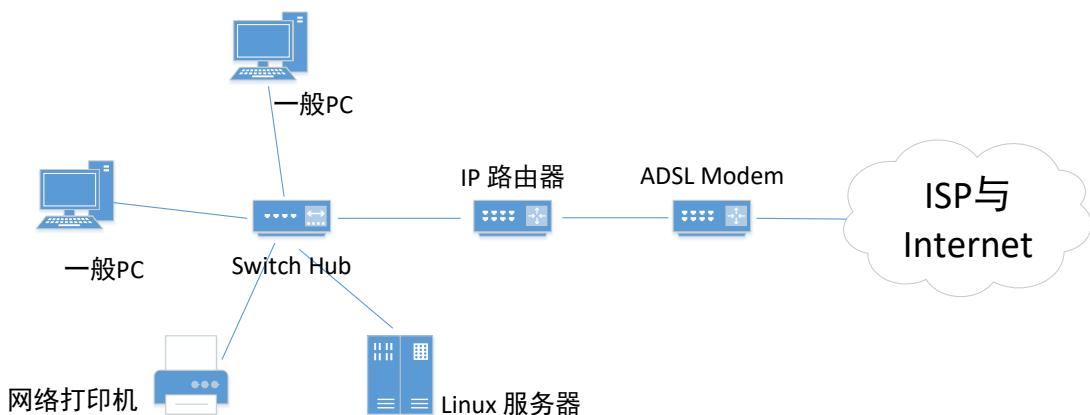


图 3-4 Linux 主机放在 LAN 里面的布线情况

3.1.2. 网络设备选购建议

1、主机硬件系统：考虑使用年限、省电、虚拟技术等

- 选购什么主机配备与该主机即将运行的服务是有关系的
- 新购买主机时，最好选有伪指令集的 CPU

2、Linux 操作系统：考虑稳定、可网络升级、能够快速取得协助支持

- 将目前的 distribution 分成两大类：一类是多功能新鲜产品，例如 Fedora；一类是强调性能稳定但软件功能较旧的企业用途产品，包括 RHEL、CentOS、SuSE 及 B2D 等
- 推荐用 RHEL/CentOS/SuSE 这几个 Linux distribution，他们比较稳定，配置不难，但里面的软件版本可能不会是最新的

3、网卡：考虑服务器用途、内置与否、驱动程序的取得

- 一般来说，目前新主机都是内置吉兆速度(Gigabit)的以太网卡，所以不需要额外购买网卡
- 不过，使用内置网卡时，需要注意到该网卡是否为特殊的网络芯片，根据经验，内置网卡的芯片通常是比较特殊的，所以可能导致 Linux 内置的网卡驱动程序无法顺利驱动该网卡，那就比较麻烦了，需要额外安装网卡驱动程序

4、Switch/Hub：考虑主机数量、传输带宽、网管功能等

- Hub 是带宽共享设备而 Switch 是具有独立带宽的非共享设备，因此从性能以及带宽的角度来看，当然是 Switch 比较好

5、网线：考虑与速度相配的等级、线缆形状、施工配线等

6、无线网络相关设备：无线网络相关设备要考虑速度、标准、安全性等

3. 2. 本书使用的内部链接网络参数与通信协议

1、新手不建议使用如图 3-4 这样的连接模式，推荐如图 3-3 这样的连接模式

3. 2. 1. 网络联机参数与通信协议

Chapter 4. 连接 Internet

4.1. Linux 连接 Internet 前的注意事项

1、如何确定网卡已被 Linux 操作系统捕捉到，即发现网卡的存在并加载了相应的网卡驱动程序

4.1.1. Linux 的网卡

1、认识网卡设备的名称

- Linux 操作系统中的各种设备几乎都以文件名来表示
- 不过，网卡(Network Interface Card, NIC)的名称确实以网卡内核模块对应的设备名称来表示的，而默认的网卡名称是 eth0，第二张网卡则为 eth1

2、关于网卡的内核模块(驱动程序)

- 网卡其实是硬件，所以当然需要内核的支持才能驱动它
- 一般来说，目前新版的 Linux distributions 默认可以支持的网卡芯片组的种类与数量已经很完备了，包括大厂的 3COM、Intel 以及低端的 RealTek、D-Link 等网卡芯片
- 如果网卡芯片组开发商不愿意提供源代码的驱动程序或者网卡太新了，使得 Linux 来不及支持时，需要通过以下方式让内核支持网卡
 - 重新编译内核
 - 编译网卡的内核模块

3、观察内核捕捉到的网卡信息

- <dmesg> | grep -in eth
- <lspci> | grep -i ethernet

4、观察网卡的模块

- <lsmod> | grep e1000
- <modinfo> e1000 <==重点就是 filename，即驱动程序放置的目录
- 这里的 e1000 仅举个例子，e1000 是 Intel 使用的模块
- ifconfig eth0 <==注意，CentOS7 默认的网卡名称不是 eth0，改动方法详见 360 浏览器收藏夹

4.1.2. 编译网卡驱动程序 (Option)

1、不建议自己编译网卡的驱动程序

2、由于编译程序需要编译程序以及内核相关信息，因此需要预安装 Gcc、Make、Kernel-Header 等软件才行

3、按照以下步骤来编译网卡驱动程序

- 取得官方网站的驱动程序
- 解压缩与编译
- 模块的测试与处理
 - 先删除旧的模块后，才能够重新加载这个模块
 - rmmod e1000
 - modprobe e1000
 - modinfo e1000
- 设定开机自动启动网卡模块(Option)

- 如果开机时能够取得这么模块，就略过该步骤
- 尝试配置 IP
- 两台虚拟机采用 NAT 的方式(不要用桥接)，直接 ping 对方的 ip 即可成功

4.1.3. Linux 网络相关配置文件

- 1、TCP/IP 的重要参数主要是 IP、Netmask、Gateway、NDS IP、主机名(Host Name)
- 2、IP 的获取方法有手动配置、DHCP 自动分配等

表格 4-1 网络参数与配置文件对应关系

| 所需网络参数 | 主要配置文件 | 重要参数 |
|----------------------------------|--|---|
| IP Netmask DHCP Gateway | /etc/sysconfig/network-scripts/ifcfg-eth0 (注意在 CentOS 或 RHEL7 下默认的文件名不是 ifcfg-eth0) | DEVICE=网卡名称 BOOTPROTO=是否使用 dhcp HWADDR=是否加入网卡 MAC 地址 IPADDR=IP 地址 NETMASK=子网掩码 ONBOOT=要不要默认启动此接口 GATEWAY=网关地址 NM_CONTROLLED=额外的网管软件，建议取消这个项目 |
| 主机名 | /etc/sysconfig/network (CentOS7 中该文件无信息) (Ubuntu16 中无该文件) | NETWORKING=要不要使用网络 NETWORKING_IPV6=是否支持 IPv6 HOSTNAME=主机名 |
| DNS IP | /etc/resolv.conf | Name Server DNS 的 IP 地址 |
| 私有 IP 对应的主机名 | /etc/hosts | 私有 IP 主机名别名 |

3、其他配置文件

- /etc/services: 这个文件是记录构建在 TCP/IP 上面的各种协议，包括 HTTP、FTP、SSH、Telnet 等服务所定义的 port number
- /etc/protocols: 定义 IP 数据包协议的相关数据，包括 ICMP/TCP/UDP 的数据包协议的定义等
- /etc/init.d/network restart <==这是一个 shell script 文件，因此在 CentOS7 下也能用，并非是对应的 systemctl *
- ifup eth0(ifdown eth0): 启动或关闭某个网络接口

4.2. 连接 Internet 的设置方法

4.2.1. 手动配置固定 IP 参数

1、固定 IP 就是指在网络参数中，只要输入既定的 IP 参数即可，一般来说它可能来源于

- 学术网：由学校单位直接给予的一组 IP 网络参数
- 固定 IP 地址的 ADSL：向 ISP 申请的一组固定 IP 的网络参数
- 企业内部或 IP 路由器内部的局域网：例如企业内部局域网使用私有 IP 时，那么 Linux 需要向企业的网关人员申请一组固定的 IP 网络参数
- **也就是说，取得的固定 IP 并非 Public，它只是一组可接受的固定 IP**

2、要修改的四个文件与相关的启动脚本，以及重新启动后需要使用什么指令查看见下表

表格 4-2 相关参数的配置文件与启动脚本及指令

| 修改的参数 | 配置文件与重要启动脚本 | 查看结果的命令 |
|---------|--|--|
| IP 相关参数 | /etc/sysconfig/network-scripts/ifcfg-eth0 /etc/init.d/network restart | ifconfig (IP/Netmask) route -n(gateway) |
| DNS | /etc/resolv.conf | dig www.google.com |
| 主机名 | /etc/sysconfig/network /etc/hosts | hostname (主机名) ping \$(hostname) reboot |

3、IP/Netmask/Gateway 的配置、启动与查看

- 设定网络参数需要修改/etc/sysconfig/network-scripts/ifcfg-eth0
- 该 ifcfg-eth0 与文件内 DEVICE 名称定义需相同
- 在这个文件内的所有配置，基本上就是 bash 的变量定义规则
- DEVICE：这个配置值后面接的设备名称要与文件名(ifcfg-eth0)那个设备名称相同，否则可能会出现设备名找不到的问题(**CentOS7 和 RHEL7 的默认设备名已经不是 eth0，通过修正后，DEVICE 与文件名并不同，但是多了个 NAME 参数，即 NAME 与文件名都改为 eth0，而 DEVICE 却是 eno16***)
- IPADDR：IP 地址，手动设置时，这就是 IP 地址
- BOOTPROTO：启动该网络接口时，使用何种协议
 - 如果是手动配置 IP 环境，就输入 static 或 none
 - 自动分配 IP 的环境中，输入 dhcp
- GATEWAY：代表的是整个主机系统的 Default Gateway
- GATEWAYDEV：如果不是使用固定 IP 作为 Gateway，而是使用网络设备座位 Gateway(通常 Router 常有这种配置)
- HWADR：网卡的 MAC 地址
 - 仅有一张网卡的情况下，这个配置值没有什么作用，可以忽略
 - 如果主机上有两张一模一样的网卡，使用的模块是相同的，此时 Linux 很可能将 eth0、eth1 搞混，造成网络配置的问题

4、DNS 服务器 IP 的定义与查看

- /etc/resolv.conf 很重要，它会影响你是否可以查询到主机名与 IP 的映射
- 参数解释
 - **nameserver**：定义 DNS 服务器的 IP 地址
 - nameserver 表明 DNS 服务器的 IP 地址，最主要的关键字
 - 如果没指定 nameserver 就找不到 DNS 服务器
 - 可以有很多行的 nameserver，在查询时就按 nameserver 在本文件中的次序进行，且只有当第一个 nameserver 没有反应时才查询下面的 nameserver
 - **domain**：定义本地域名
 - 很多程序用到它，如邮件系统
 - 当为没有域名的主机进行 DNS 查询时，也要用到。如果没有域名，主机名将被使用，所得的域名为：**删除主机名在第一个点"."前面的内容**
 - **search**：定义域名的搜索列表，它的多个参数指明域名查询次序

- 当要查询没有域名的主机，主机将在由 `search` 声明的域中分别查找
- `domain` 和 `search` 不能共存，如果同时存在，后面出现的将被使用
- **`sortlist`**: 允许将得到域名进行特定的排序
- <dig> 域名的解析
- 中国大陆可用的 DNS IP: 114.114.114.114
- `/etc/resolv.conf` 例子
 - nameserver 114.114.114.114
- CentOS7 上(192.168.136.130)该文件的内容
 - search localdomain <=这个什么意思
 - nameserver 192.168.136.2 <=为什么是这个地址
 - Windows 的 IP 为 192.168.136.1,
 - 在 Windows 架设的 Linux 虚拟机 IP 为 192.168.136.130
 - 那么这个 192.168.136.2 什么意思？是 NAT 模式下一个虚拟的 NAT 主机吗
 - 这个文件的内容是依据 VM 的虚拟机 DHCP 服务器的配置文件写入的
- CentOS7 上(192.168.136.130)该文件的内容可以改成如下格式
 - nameserver 114.114.114.114

5、主机名的修改、启动与查看<hostname><localhost>

- 主机名修改就需要改`/etc/sysconfig/network` 以及`/etc/hosts` 这两个文件
 - `/etc/sysconfig/network` 增加以下两行
 - `NETWORKING=yes`
 - `HOSTNAME=<主机名>`
 - `/etc/hosts` 增加以下一行
 - <IP> <主机名>
 - **重启**
 - 当修改过`/etc/sysconfig/network` 里面的 `HOSTNAME` 后，务必要重新启动 (reboot)，这个档案主要的功能在于设定『主机名称(HOSTNAME)与启动 Network 与否』
 - `/etc/hosts` 文件的作用相当于 DNS，提供主机名与 IP 的对应关系，而主机名称的提供需要靠`/etc/sysconfig/network` 来设定
 - `localhost` 是什么
 - `localhost` 一般是`/etc/hosts` 中配置的 127.0.0.1 的别名
 - Mac 平台下
 - `sudo scutil --set HostName <你的主机名>`
 - 配置文件"/Library/Preferences/SystemConfiguration//preferences.plist"
 - Ubuntu16 平台下
 - `hostname <你的主机名>`
 - 配置文件：没找到
 - 问题：
 - `/etc/hostname` 什么作用
 - 当没有修改`/etc/sysconfig/network` 以及`/etc/hosts` 时，命令 `hostname` 会打印`/etc/hostname` 的内容
 - 当修改`/etc/sysconfig/network` 以及`/etc/hosts` 时后，命令 `hostname` 又不打印`/etc/hostname` 的内容了

4.2.2. 自动取得 IP 参数(DHCP 方法, 适用 Cable Modem、IP 路由器的环境)

1、自动取得其实就是有一台主机提供 DHCP 服务给整个网络内的计算机

- 例如 IP 路由器就可能是一台 DHCP(Dynamic Host Configuration Protocol, 级动态主机配置协议)主机,

2、DHCP 连接方式

- Cable Modem: 使用有线电视网络连接的方式
- ADSL 多 IP 的 DHCP 方式
- IP 路由器或 NAT 搭建了 DHCP 服务时

4.2.3. ADSL 拨号上网(适用 ADSL 拨号以及光纤接入)

1、要拨号上网, 可以使用 rp-pppoe 这套软件来帮忙

2、rp-pppoe 使用的是 Point to Point over Ethernet 的点对点协议所产生的网络接口, 因此当顺利地拨号成功后, **会多产生一个网络接口 ppp0**

3、由于 ppp0 是构建在以太网卡的, 必须要由以太网卡, 同时即使拨号成功后, 也不能将没有用到的 eth0 关闭, 因此拨号成功会有如下接口

- 内部环回测试用的 lo 接口
- 网卡 eth0 这个接口
- 拨号之后产生的通过 ISP 对外连接的 ppp0 接口

4、关于 eth0 使用上的思考

- 这张网卡(假设是 eth0)可以连接内部网络(LAN)
 - ppp0 可以连上 Internet, 但是内网则使用 eth0 来跟其他内部主机连接时, 那么你的 IP 配置参数/etc/sysconfig/network-scripts/ifcfg-eth0 应该要给予一个私有 IP 以便内部的 LAN 也可以通过 eth0 来进行连接
 - **千万不要定义 GATEWAY, 因为 ppp0 拨号成功后, ISP 会主动给予 ppp0 接口一个可以连上 Internet 的 Default Gateway, 如过你又定义另一个 Default Gateway, 两个网关可能会造成网络不通**
- 这台主机仅连接 ADSL 调制解调器, 并没有接入内部网络
 - /etc/sysconfig/network-scripts/ifcfg-eth0 中" ONBOOT=no"即可
 - 拨号启动 ppp0 会自动唤醒 eth0, 只是 eth0 不会有 IP 信息

4.3. 无线网络——以笔记本电脑为例

1、除了使用 RJ-45 接口线路来连接网络意外, 由于笔记本电脑渐渐广为使用, 因此在笔记本电脑上面的无线网络(Wireless Local Area Network, WLAN)也越来越重要

2、无线网络机制非常多, 常听到的 Wi-Fi(可以理解为 802.11 相关标准)以及 WiMAX(802.16)等

4.3.1. 无线网络所需要的硬件: AP、无线网卡

1、在 RJ-45 接口的以太网环境中, 以 Switch/Hub 以及网卡与网络线最重要, 该架构中主要以 Switch/Hub 连接所有的网络设备

2、在无线网络中, 也需要一个接受信号的设备, 那就是**无线接入点(Wireless Access Point)**, 另一个设备就是**无线网卡**

3、无线接入点本身就是一个 IP 路由器，它本身会有两个接口，一个可以与外部的 IP 做沟通，另外一个则是作为 LAN 内部其他主机的 GATEWAY

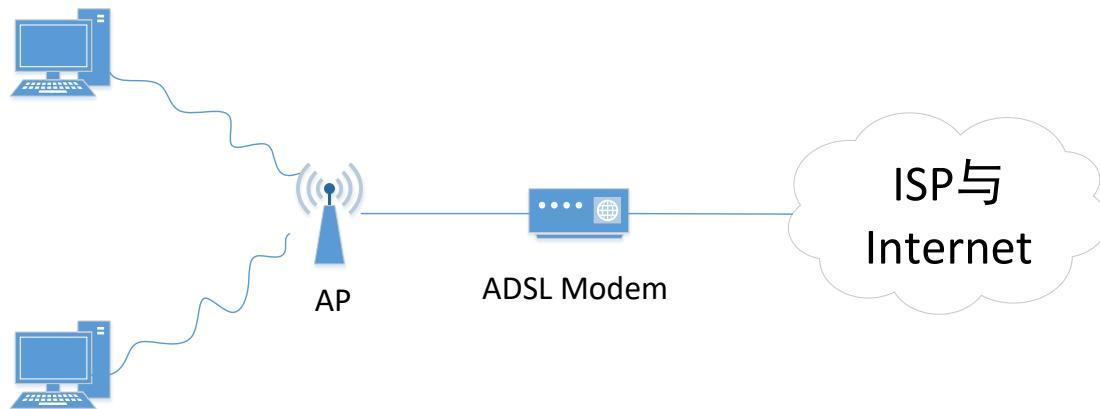


图 4-1 无线网络的连接图示

4.3.2. 关于 AP 的设置：网络安全方面

1、如果 AP 不设置任何连接限制，那任何拥有无线网卡的主机都可以通过这个 AP 连接上你的 LAN

2、通常，我们都会认为 LAN 是可信任的网络，所以内部没有防火墙，即不设防状态

3、**无线网络的安全性一定具有很大的漏洞**，因为无线网络的传输并不是通过实体的网线，而是通过无线信号，实体网线很好控制，无线信号却不好控制

4、一般可以进行如下限制

- 在 AP 上面使用网卡卡号(MAC)来作为是否可以访问 AP 的限制
 - 如此一来，只有你允许的网卡才能够访问 AP，自然很安全
 - 当有其他主机想要通过 AP 连接 Internet 时，就需要手动登录 AP 去进行 MAC 的设置，在经常有移动设备的环境中，这个方法比较麻烦
- 设置你的 AP 连接加密机制与秘钥

5、ESSID/SSID

- 每台 AP 都会有一个联机的名字，那就是 SSID 或 ESSID
- SSID 可以提供给 Client 端，当 Client 端进行无线连接时，它必须说明它要利用哪一台 AP，ESSID 就是那时需要输入的数据

4.3.3. 利用无线网卡开始连接

1、无线网卡有很多模式，想要知道是否捕捉到这张网卡，可以使用 lsusb 来检查

Chapter 5. Linux 中常用的网络命令

5.1. 设置网络参数的命令

1、概览

- ifconfig: 查询、设置网卡与 IP 网络相关参数
- ifup,ifdown: 这是两个 script 文件, 其作用是通过更简单的方式来启动与关闭网络接口
- route: 查看、配置路由表
- ip: 整合式的命令, 可以直接修改上述提到的功能

5.1.1. 手动/自动配置 IP 参数与启动/关闭网络接口: ifconfig, ifup, ifdown

1、这三个命令的用途都是启动网络接口

- 不过 ifup 与 ifdown 仅能就/etc/sysconfig/network-scripts 内的 ifcfg-ethX 进行启动或关闭的操作, 并不能直接修改网络参数, 除非手动调整 ifcfg-ethX 文件才行
- ifconfig 则可以直接手动为某个接口配置 IP 或调整网络参数

2、<ifconfig>

- ifconfig 主要是可以手动启动、查看与修改网络接口的相关参数, 可以修改的参数很多, 包括 IP 参数以及 MTU 等
- ifconfig [interface] [up|down]
- ifconfig [interface] [options]
- interface: 网卡接口名称, 例如 eth0, eno16*等
- options: 可用参数, 包括
 - up,down: 启动(up)或关闭(down)该网络接口(不涉及任何参数)
 - mtu: 可以设置不同的 MTU 数值, 例如 mtu1500
 - netmask: 子网掩码
 - broadcast: 广播地址
- 查询参数意义详解
 - eth0: 网卡代号, 或者是 lo 这个 loopback
 - HWaddr: 网卡硬件地址, **Centos 里面显示的是 ether**
 - inet addr/inet: IPv4 的 IP 地址, 后续的 Bcast(broadcast)、Mask(netmask) 分别代表 Broadcast 与 Netmask
 - inet6 addr: IPv6 版本的 IP 地址
 - MTU: 最大传输单元
 - RX: 网络启动到目前为止数据包接受情况
 - TX: 网络启动到目前为止数据包发送情况
- 例子
 - ifconfig eth0 192.168.100.100 <=修改 IP 地址, 会自动更新 netmask, network, 以及 broadcast 等参数
 - ifconfig eth0 192.168.100.100 netmask 255.255.255.128 mtu 8000 <=手动设置其余参数, 同时设置 MTU 参数
 - ifconfig eth0:0 192.168.50.50 <=在该实体网卡上, 再仿真一个网络接口, 即在一张网卡上设置多个 IP 的意思

- ifconfig eth0:0 down <==关掉这个接口
- /etc/init.d/network restart <==以 ifcfg-ethX 的设置为主

3、<ifup><ifdown>

- 实时手动修改网络接口参数，可以利用 ifconfig 实现，如果要直接以配置文件，也就是/etc/sysconfig/network-scripts 里面的 ifcfg-ethX 等文件的设置参数来启动网络接口的话，需要通过 ifup 或 ifdown 来完成
- ifup eth0 <==它会直接到/etc/sysconfig/network-scripts 目录下查找对应的配置文件
- 如果以 ifconfig eth0 的方式来设置或者修改了网络接口后，那就无法再以 ifdown eth0 来关闭了，因为 ifdown 会分析对比当前网络参数与 ifcfg-eth0 是否相符，不相符就放弃本次操作
- 因此，当 ifconfig 修改完毕后，应该要用 ifconfig eth0 down 来关闭接口

5.1.2. 修改路由：route

1、<route>

- route [-nee]
- route add [-net|-host] [网络或主机] netmask [mask] [gw|dev]
- route del [-net|-host] [网络或主机] netmask [mask] [gw|dev]
- -n: 不要使用通信协议或主机名，直接使用 IP 或 port number，即在默认情况下，route 会解析出该 IP 的主机名，若解析不到则会有延迟，因此一般加上这个参数
- -ee: 显示更详细的信息
- -net: 表示后面接的路由为一个网络
- -host: 表示后面接的为连接到单个主机的路由
- netmask: 与网络有关，可设置 netmask 决定网络的大小
- gw: gateway 的缩写，后接 IP 数值
- dev: 如果只是要制定由哪块网卡连接出去，则使用这个设置，后接网卡名，例如 eth0 等
- 显示参数详解
 - Destination、Genmask: 这两个参数就分别是 network 与 netmask
 - Gateway: 该网络通过哪个 Gateway 连接出去，若显示 0.0.0.0(default) 表示该路由直接由本级传送，也就是通过局域网的 MAC 直接传送，如果显示 IP 的话，表示该路由需要经过路由器(网关)的帮助才能发送出去
 - Flags:
 - U(route is up): 该路由是启动的
 - H(target is a host): 目标是一台主机而非网络
 - G(use gateway): 需要通过外部的主机来传递数据包
 - R(reinstate route for dynamic routing): 使用动态路由时，恢复路由信息的标志
 - D(Dynamically installed by daemon or redirect): 动态路由
 - M(modified from routing daemon or redirect): 路由已经被修改了
 - ! (reject route): 这个路由将不会被接受
 - Iface: 该路由传递数据包的接口
- 路由排序依序由小网络到大网络，最后是默认路由，即先匹配小网络，若

匹配则直接通过 Iface 传输出去，若不匹配则尝试匹配下一个网络，直到默认路由

- 删除路由时，需要将路由表上的信息都写入
 - route del -net 169.254.0.0 netmask 255.255.0.0 dev eth0

5.1.3. 网络参数综合命令: <ip>

1、基本上 ip 综合了 ifconfig 与 route 两个命令

2、关于接口设备(device)的相关设置: <ip link>

- ip [-s] link show <==单纯地查看该设备的相关信息
- ip link set [device] [动作参数]
- show: 仅列出这个设备的相关属性，如果加上-s 会显示更多统计信息
 - ip -s link show eth0
- set: 可以开始设置项目，device 指的是 eth0、eth1 等设备名称
- 动作参数包括
 - up|down: 启动(up)或关闭(down)某个接口，其他参数使用默认的以太网
 - address: 如果这个设备可以更改 MAC 的话，用这个参数修改
 - name: 给予这个设备一个特殊的名字
 - mtu: 最大传输单元
- set 例子
 - ip link set eth0 up
 - ip link set eth0 down
 - ip link set eth0 mtu 1000
- 更新网卡的 MTU 使用 ifconfig 可以实现，**但是更改网卡名称、MAC 地址的信息，就得使用 ip 了**，不过得先关闭该网卡，否则不会成功
 - 对于那些使用配置文件启动的网卡来说，改动网卡名称并不会使对应的配置文件也改动哦

3、关于**额外 IP**的相关设定: <ip address>

- 如果说 ip link 与 OSI 七层协议的第二层数据链路层有关，那么 ip address 就与第三层网络层有关了
- ip address 主要设置与 IP 有关的各项参数，包括 netmask、broadcast 等
- ip address show <==查看 IP 参数
- ip address [add|del] [IP 参数] [dev 设备] [相关参数]
- show: 仅显示接口的 IP 信息
- add|del: 进行相关参数的增加(add)或删除(del)设置
 - IP 参数: 主要就是网络的设置，例如 192.168.100.100 之类的设置
 - dev: 这个 IP 参数所要设置的接口，例如 eth0, eth1 等
 - 相关参数
 - broadcast: 设置广播地址，如果设置值为 "+" 表示让系统自动计算
 - label: 设备的别名
 - scope:
 - ◆ global: 允许来自所有来源的连接
 - ◆ site: 仅支持 IPv6，仅允许本地主机的连接
 - ◆ link: 仅允许本设备自我连接
 - ◆ host: 仅允许本主机内部的连接

◆ 默认使用 global

➤ 例子

- ip address add 192.168.50.50/24 broadcast + dev eth0 label eth0:liuye
- ip address del 192.168.50.50/24 dev eth0

4、路由相关设定: <ip route>

- 路由的查看与设定, ip route 与 route 功能差不多
- ip route show <=>单纯显示路由的设置
- ip route [add|del] [IP 或网络号] [via gateway] [dev 设备]
- show: 单纯地显示出路由表, 也可以使用 list
- add|del: 添加或删除路由
 - IP 或网络: 可使用 192.168.50.0/24 之类的网络号或者是单纯的 IP 地址
 - via: 从哪个 gateway 出去, 不一定需要
 - dev: 由哪个设备连接出去, 不一定需要
 - mtu: 可以额外设置 mtu 值

5. 1. 4. 无线网络: iwlist, iwconfig

1、<未完成>: 没有上述两个命令

5. 1. 5. DHCP 客户端命令: dhclient

1、如果使用 DHCP 协议在局域网内获取 IP, 除了编辑 ifcfg-eth0 内的 BOOTPROTO, 更快的做法是利用 dhclient 这个命令

➤ dhclient eth0

5. 2. 网络排错与查看命令

5. 2. 1. 两台主机的两点沟通: ping

1、ping 主要通过 ICMP 数据包来进行整个网络的状态报告

- 最重要的就是 ICMP type0、8 这两个类型, 分别是要求回送与主动回送网络状态是否存在的特性
- ping 需要通过 IP 数据包来传送 ICMP 数据包
- IP 数据包里面有个相当重要的 TTL 属性

2、<ping>

- ping [选项与参数] IP
- -c 数值: 后面接执行 ping 的次数
- -n: 在输出数据时不进行 IP 与主机名的反查, 直接使用 IP 输出(速度快)
- -s 数值: 发送出去的 IMCP 数据包大小, 默认 56bytes
- -t 数值: TTL 值, 默认 255, 每经过一个节点就会少 1
- -W 数值: 等待响应对方主机的秒数
- -M [do|dont]: 主要在检测网络 MTU 数值大小
 - do: 代表传送一个 DF(Don't Fragment)标志, 让数据包不能重新拆包与打包
 - dont: 代表不要传送 DF 标志, 表示数据包可以在其他主机上拆包与打包

3、修改网卡的 MTU 值可以通过 ifconfig 或 ip 命令来完成，而跟踪整个网络传输的最大 MTU 时，最简单的方法就是通过 ping 发送一个大数据包，并且不许中继路由器或 Switch 将该数据包重组，就可以处理了

4、MTU 不要随便调整，除非真的有问题

5、常见各种接口的 MTU 值

- Ethernet: 1500
- PPPoE: 1492
- Dial-up(Modem): 576
- 注意，IP 数据包包头(不含 options)占用了 20bytes(详见图 2-4)，加上 ICMP 报头 8bit，因此当使用-s size 时，那个数据包的大小就需要扣除 28bits，例如 MTU 为 1500 时，下达的命令为"ping -s 1472 -M do <ip>"

5.2.2. 两台主机各节点分析：traceroute

1、ping 是两台主机之间连通性判断，traceroute 可以跟踪两台主机之间通过的各个节点的通信状况的好坏

2、<traceroute>

- traceroute [选项与参数] IP
- -n: 可以不必进行主机的名称解析，单纯用 IP，速度较快
- -U: 使用 UDP 的 port 33434 来进行检测，这是默认的检测协议
- -I: 使用 ICMP 的方式来进行检测
- -T: 使用 TCP 来进行检测，一般使用 port 80 检测
- -w: 若对方主机在几秒钟内没有回应就声明不通，默认 5 秒
- -p: 端口号
- -i 设备：用在比较复杂的环境，如果网络接口很多很复杂时，才会用到这个参数

3、traceroute 会针对欲连接的目标的所有 node 进行 UDP 的超时等待

- 会针对每个节点做 UDP 的响应等待，并检测回复的事件，每节点检测三次
- 返回星号代表 node 可能没有某些防护措施，让我们发送的数据包信息被丢弃掉，由于我们直接通过路由器传递数据包，并没有进入路由器获取路由器的使用资源，所以某些路由器仅支持数据包传递，并不会接受来自客户端的各项检测，于是出现该问题

5.2.3. 查看本机的网络连接与后门：netstat

1、某个网络服务明明已经启动了，但就是无法进行连接，首先要查询一下网络接口所监听的端口(port)，来看看是否已经真的启动

2、<netstat>

- netstat -[rn] <=与路由有关的参数
- -r: 列出路由表(route table)，功能如同 route
- -n: 不使用主机名与服务名称，使用 IP 与 port number，如同 route -n
- netstat -[antulpc] <=与网络接口有关的参数
- -a: 列出所有的连接状态，包括 tcp/udp/unix socket 等
- -t: 仅列出 TCP 数据包的连接
- -u: 仅列出 UDP 数据包的连接
- -l: 仅列出已在 Listen(监听)的服务的网络状态

- -p: 列出 PID 与 Program 的文件名
- -c: 可以设置几秒种后自动更新一次, 例如-c 5 为每 5s 更新一次网络状态的显示
- 显示参数说明
 - Proto: 该连接的数据包协议, 主要为 TCP/UDP 等数据包
 - Recv-Q: 非用户程序连接所复制而来的总 byte 数
 - Send-Q: 由远程主机发送而来, 但不具有 ACK 标志的总 byte 数, 亦指主动连接 SYN 或其他标志的数据包所占的 byte 数
 - Local Address: 本地端的地址, 可以使 IP, 也可以是完整的主机名, 使用的格式是"IP:port"
 - Foreign Address: 远程主机 IP 与 port number
 - stat: 状态栏
 - ESTABLISHED: 已建立连接的状态
 - SYN_SENT: 发出主动连接(SYN 标志)的连接数据包
 - SYN_RECV: 接收到一个要求连接的主动连接数据包
 - FIN_WAIT1: 该套接字服务已中断, 该连接正在断线当中
 - FIN_WAIT2: 该连接已挂断, 但正在等待对方主机响应断线确认的数据包中
 - TIME_WAIT: 该连接已挂断, 但 socket 还在网络上等待结束
 - LISTEN: 通常在服务的监听 port, 可以使用-l 参数查阅

3、netstat 的功能就是查看网络的连接状态, 而网络连接状态中, 又以"我目前开了多少 port 在等待客户端的连接"以及"目前我的网络连接状态中, 有多少连接已建立或产生问题"最常见

5. 2. 4. 检测主机名与 IP 的对应: host、nslookup

1、<host>

- 用来查出某个主机名的 IP
- host [-a] hostname [server]
- -a: 列出该主机详细的各项主机名设置数据
- [server]: 可以使用不是由/etc/resolv.conf 文件定义的 DNS 服务器的 IP 来查询

2、<nslookup>

- 用来作为 IP 与主机名对应的检查, 使用的是/etc/resolv.conf 这个文件来作为 DNS 服务器的来源选择
- nslookup [-query=[type]] [hostname|IP]
- -query=type: 查询的类型, 除了传统的 IP 与主机名对应外, DNS 还有很多信息, 包括 mx, cname 等
- 目前大家都建议使用 dig 这个命令来渠道 nslookup

5. 3. 远程连接命令与即时通信软件

5. 3. 1. 终端机与 BBS 连接: telnet

1、telnet 是早期个人计算机上面连接到服务器工作时, 最重要的一个软件

- 不但可以直接连接到服务器上面，还可以用来连接 BBS
- telnet 本身的数据在传输过程中使用的是明文(原始数据，没有加密)
- telnet 还可以用来连接到某个 port(服务)上

5.3.2. FTP 连接软件: ftp、lftp

- 1、文字接口的 FTP 软件主要有 ftp、lftp 两个
- 2、图形接口的 FTP 软件，在 CentOS 上默认有 gftp
- 3、<ftp>
 - ftp [host|IP] [port]
- 4、<lftp>
 - 自动化脚本：单纯使用 ftp 总是觉得麻烦，lftp 默认使用匿名登录 FTP 服务器，可以使用类似网址的方式取得数据，比单纯的 ftp 要好用一些
 - lftp [-p port] [-u [user[,pass]]] [host|IP]
 - lftp -f filename
 - lftp -c "commands"
 - -p: 后面可以直接连接上远程 FTP 主机提供的 port
 - -u: 后面则是接上账号与密码，就能够连接上远程主机了，如果没有加账号密码，lftp 默认会使用 anonymous 尝试匿名登录
 - -f: 可以将命令写入到脚本中，这样可以帮助进行 shell script 自动处理
 - -c: 后面加上所需要的命令

5.3.3. 图形接口的即时通信软件: Pidgin(gaim 的延伸)

- 1、<未完成>：什么鬼

5.4. 文字接口网页浏览

- 1、在文字接口下上网浏览的工具：links 以及 wget

5.4.1. 文字浏览器: links

- 1、<links>
 - **最大的功能：查阅 Linux 本机上面 HTML 语法写成的文件数据**
 - links [options] [URL]
 - -anonymous [0|1]: 是否使用匿名登录的意思
 - -dump [0|1]: 是否将网页的数据直接输入到 standard out 而非 links 软件功能
 - links -dump <http://WWW.yahoo.com> > yahoo.html
 - -dump_charset: 后面接想要通过 dump 输出到屏幕的语系编码，简体中文使用 cp936
 - 进入后一些常见功能按键
 - h: history，曾经浏览过的 URL 就显示到画面中
 - g: Goto URL，按 g 后输入网页地址(RUL)
 - d: download，将该链接数据下载到本机成为文件
 - q: Quit，离开 links 这个软件
 - o: Option，进入功能参数的设置值修改中，最终可写入/.elinks/elinks.conf

中

5.4.2. 文字接口下载器: wget

1、<wget>

- 用于网页数据的取得，例如 Linux 的核心是放置在 www.kernel.org 内，主要同时提供 FTP 与 HTTP 来下载，我们可以通过 lftp 来下载资料，也可以用 wget 通过浏览器来下载
- wget [option] [网址]
- --http-user=username
- --http-password=password
- --quiet: 不要显示 wget 在捕获数据时显示信息

5.5. 数据包捕获功能

1、分析数据包的流向：通过分析数据包的流向，我们可以了解一条连接应该是如何进行双向的连接的操作，也就会清楚地了解到可能发生的问题所在

5.5.1. 文字接口的数据包捕获器：tcpdump

1、tcpdump 可以说其实就是一个黑客软件，因为它可以分析数据包的流向，连数据包的内容也可以进行监听

2、<tcpdump>

- tcpdump [-AennqX] [-i 接口] [-w 存储文件名] [-c 次数] [-r 文件] [所要摘取数据包的格式]
- -A: 数据包的内容以 ASCII 显示，通常用来抓取 WWW 的网页数据包数据
- -e: 使用数据链路层(OSI 第二层)的 MAC 数据包数据来显示
- -nn: 直接以 IP 以及 port number 显示，而非主机名与服务名称
- -q: 仅列出较为简短的数据包信息，每一行的内容比较精简
- -X: 可以列出十六进制(hex)以及 ASCII 的数据包内容，对于监听数据包很有用
- -i: 后面接要监听的网络接口，例如 eth0 等
- -w: 将监听得到的数据包存储下来
- -r: 从后面接的文件将数据包数据读出来
- -c: 监听数据包数，没有这个参数则会一直监听，直到[ctrl]+C

5.5.2. 图形接口数据包捕获器：wireshark

1、CentOS 下安装方式: yum install wireshark wireshark-gnome

2、显示内容

- 第一区块主要显示的是数据包的报头数据，内容有点类似 tcpdump 的显示结果
- 第二块区域则是详细的报头数据，包括数据帧、通信协议内容以及 Socket pair 等信息
- 第三区块则是十六进制与 ASCII 码的显示结果

5.5.3. 任意启动 TCP\UDP 数据包的端口连接: nc、netcat

1、<nc>

- nc [-u] [IP|host] [port]
- nc -l [IP|host] [port]
- -l: 作为监听之用，也就是打开一个 port 来监听用户的连接
- -u: 不使用 TCP 而是使用 UDP 作为连接的数据包状态

Chapter 6. Linux 网络排错

6.1. 无法连接网络的原因分析

1、网络时通时不通的问题，主要可以归类为硬件问题与软件设置问题

- 硬件问题比较麻烦，需要通过一些专门的设备来分析硬件
- 至于软件，绝大部分都是设置错误或者观念错误而已

6.1.1. 硬件问题：网线、网络设备、网络布线等

1、网线问题

- 网线分成直连线和交叉线(RJ-45)，并不是所有设备都支持自动分辨交叉线与直连线功能的
- 网线被截断
- 网线过度扭曲变形造成信号不良
- 资质网络接头(RJ-45 跳线头)品质不良
- 网络接头与设备(Hub)接触不良

2、网卡、Hub 及 Router 等网络设备的问题

- 网卡不稳定，质量不佳，或者与整体系统的兼容性不好
- 各网络设备接头质量不佳，接触不良，造成信号衰减
- 由于网络设备所在环境恶劣导致的宕机问题
- 各网络设备使用方法不良，造成设备功能衰减

3、设备配置规则

- 最容易发生的就是太长的网线会造成信号的衰减，导致网络连接的时间太长甚至无法连接
- 其他网络节点配置问题
 - 使用错误的网线，最长发生在直连线与交叉线中(现在这个问题比较少了)
 - 架设网线过长，导致信号衰减太严重
 - 其他噪声的干扰，网线或者网络设备旁边有太强的电磁场
 - 局域网上面，节点或者其他设备太多

6.1.2. 软件问题：IP 参数设置、路由设置、服务器与防火墙等

1、网卡的 IP/Netmask 设置错误

- 同一个 IP 在同一个网段中出现 IP 冲突
- 子网掩码设置错误
- 网卡驱动程序使用错误
- 网卡的 IRQ 和 I/O Address 设置冲突

2、路由问题(Route Table)

- 最常见的就是默认路由(Default Gateway)设置错误
- 或者因路由接口不符导致的问题，使得数据包没有办法顺利地发送出去

3、通信协议不相符

4、网络负荷问题>Loading)

- 当同时有大量数据包涌进 Server、Hub 或者一个网络中时，就有可能造成网络的停顿甚至故障
- 如果局域网有人使用 BT(P2P)软件或者有人中毒导致蠕虫充满整个局域网，

也会造成网络停顿的问题

5、其他问题

- 例如一些 Port 被防火墙挡住了，造成无法执行某些网络资源
- 应用程序本身 Bug 问题
- 应用程序中用户的网络设置错误
- 不同的操作系统的兼容性问题

6.1.3. 问题的处理

1、两个原则

- 先从自身的环境开始检测
 - 从 PC 上的网卡查起，再到网络，Hub、调制解调器等硬件
 - 这个步骤当中，最好用的软件就是 ping，而你最好拥有两台以上的主机来进行连接测试
- 确定硬件没问题后，再来思考软件的设置问题

2、处理步骤

- **了解问题：**
 - 这个问题是刚发生的还是之前做了什么动作而导致无法连接
- **确认 IP：**
 - 先看看自己网卡有无驱动，能否取得正确的 IP 相关参数来连接
- **确认局域网连接：**
 - 利用 ping 来沟通两台主机(或路由器)，确定网络与中继器的 Hub/Switch 工作是否正常
- **确认对外连接：**
 - 看主机或 IP 路由器能否顺利取得 IP 参数，并以 ping 的方法确定对外连接是否可以成功
 - 例如 ping 168.95.1.1 / ping www.baidu.com 等
- **确认 DNS 查询：**
 - 利用 nslookup、host 或者 dig 检查 www.baidu.com
- **确认 Internet 节点：**
 - 利用 traceroute 检查各节点是否有问题
- **确认对方服务器：**
 - 是否服务器太忙了
 - 或它的机器宕机了
- **确认我方服务器：**如果是别人连不上我这台服务器
 - 那检查主机的某些服务是否正确启动，可利用 netstat 检查
 - 是否某些安全机制的软件没有设置好
- **防火墙权限的问题：**
 - 是否由于权限设置错误所致
 - 是否由于你的机器有防火墙忘记启动可连接的端口所致，可以通过 tcpdump 来处理

6.2. 处理流程

6.2.1. 步骤 1：网卡工作确认

1、确认网卡已经驱动成功

- 利用 `lspci` 以及 `dmesg` 来查询相关的设备与模块的对应

2、确定可以手动直接建立 IP 参数

- 在顺利加载网卡的模块，并且取得网卡代号之后，可以利用 `ifconfig` 或 `ip` 来直接给予网卡一个网络地址
- 操作
 - `ifconfig eno1677736 192.168.1.100 <==`随便给予一个地址
 - `ping 192.168.1.100 <==`看是否可以 ping 通
- 如果有响应的话，说明网卡的设置应该没有问题了，接下来看局域网内各个硬件了

6.2.2. 步骤 2：局域网内各项连接设备检测

1、关于网段的概念

- 能否成功架设局域网，与网段的概念有关，得明白 `192.168.1.0/24` 这种网络的表达方式的意义
- 而且子网掩码的意义也得了解

2、关于 Gateway 与 DNS 的设置

- `Gateway` 与 `DNS` 最容易被搞混，这两个并非是填写你的 Linux 主机的 IP，应该是 `Gateway` 与 IP 路由器(或 NAT 主机)的 IP，在 `DNS` 的 IP 设置中填写 `168.95.1.1`

3、关于 Windows 端的工作组与计算机名称

- 假如你还需要资源共享，那么你就必须在 Windows 系统中开放文件共享，并且建议所有计算机将“工作组”设置相同，但“计算机名称”则不能相同，不过，这个只与网上邻居以及 SAMBA 服务器有关

4、假设局域网内所有主机的 IP 都设置正确了，那么接下来你就可以使用 `ping` 来测试两个局域网主机的连接，这个连接的动作可以让你测试两台主机间的各项设备，包括网络、Hub/Switch 等，如果不成功，就了解以下几项设置

- IP 参数是否设置正确
 - 先确定 IP/Netmask 是对的
- 连接的线缆问题
 - 判断每台主机是否顺利连接到 Hub/Switch 最简单的方法就是通过连接到 Switch 上的灯来判断
- 网卡或 Hub/Switch 本身出问题

6.2.3. 步骤 3：取得正确的 IP 参数

1、正确的 IP 参数：**要连接上 Internet 必须以跟 Public IP 进行沟通才行，而与 Public IP 取得沟通的方法(并没有说取得的 IP 就是 Public IP 哟)，比较常见的有 ADSL、Cable Modem、学术网、电话拨号等**

- 在 CentOS 中，我们可以通过修改 `/etc/sysconfig/network-scripts/ifcfg-eth0` 或者利用 `rp-pppoe` 来进行拨号，**无论如何都得连接到某个 ISP 中去**

2、ADSL 拨号之后竟然取得一组 Private IP，导致无法搭建服务器，这样的情况是

否合理？

- 因为取得 IP 只是为了要连接到 ISP 去而已，而 ISP 与你的主机当然可以通过 Private IP 来连接
- 如果这样的话，就无法架设网站了

3、无法顺利取得 IP 最常见的错误就是 BOOTPROTO 的值设置错了，因为 static 与 DHCP 协议所产生的 IP 要求是不一样的

6.2.4. 步骤 4：确认路由表的规则

1、如果已经取得正确的 IP 参数的话，那么接下来就是测试一下是否可以连接上 Internet

- 建议可以尝试使用 ping 来连接 Hinet 的 DNS 主机，也就是 168.95.1.1
- 如果有响应，就表示你的网络基本上已经没有问题了，可以了解到 Internet
- 如果你的 Public IP 无法连接到外部(例如 168.95.1.1)，可能的问题就出在路由与防火墙上面了，若没有启动防火墙，那么问题就出现在路由上面

6.2.5. 步骤 5：主机名与 IP 查询的 DNS 错误

1、如果可以 ping 到 168.95.1.1 这个 Internet 上面的主机，却无法使用浏览器在地址栏浏览 www.google.com 的话，那 99% 以上的问题来自于 DNS 解析的困扰

- 查看/etc/resolv.conf
- 最常见的错误是 nameserver 的字母拼写错了
- 如果 Client 端是 Window 系统，那么 Windows 端的 DNS 设置与主机端的 /etc/resolv.conf 的内容相同即可
- 另外，**每一台主机都会有主机名，默认的主机名是 localhost**，这个主机名会有一个 127.0.0.1 的 IP 对应在/etc/hosts 当中
 - **当你修改过主机名，该主机名却无法有一个正确的 IP 的话，那么你的主机在开机时，会有好几十分钟的延迟，所以/etc/hosts 与你的主机名对应，对于内部私有网络来说，是相当重要的设置项目**

6.2.6. 步骤 6：Linux 的 NAT 服务器或 IP 路由器出问题

1、NAT 服务器最简单的功能就是 IP 路由器

- **NAT 主机一定是台路由器，所以你必须要在 Linux 上面查看好正确的路由信息(???啥意思，与虚拟机的设置选项 NAT 有什么关系???)**
- NAT 主机上面的防火墙设置是否合理，IP 路由器上面是否设置过滤的机制等，都会影响到对外连接是否能够成功的关键点

6.2.7. 步骤 7：Internet 的问题

6.2.8. 步骤 8：服务器的问题

- 1、服务器并没有开放该项服务
- 2、主机的权限设置错误
- 3、安全机制设置错误
- 4、防火墙问题

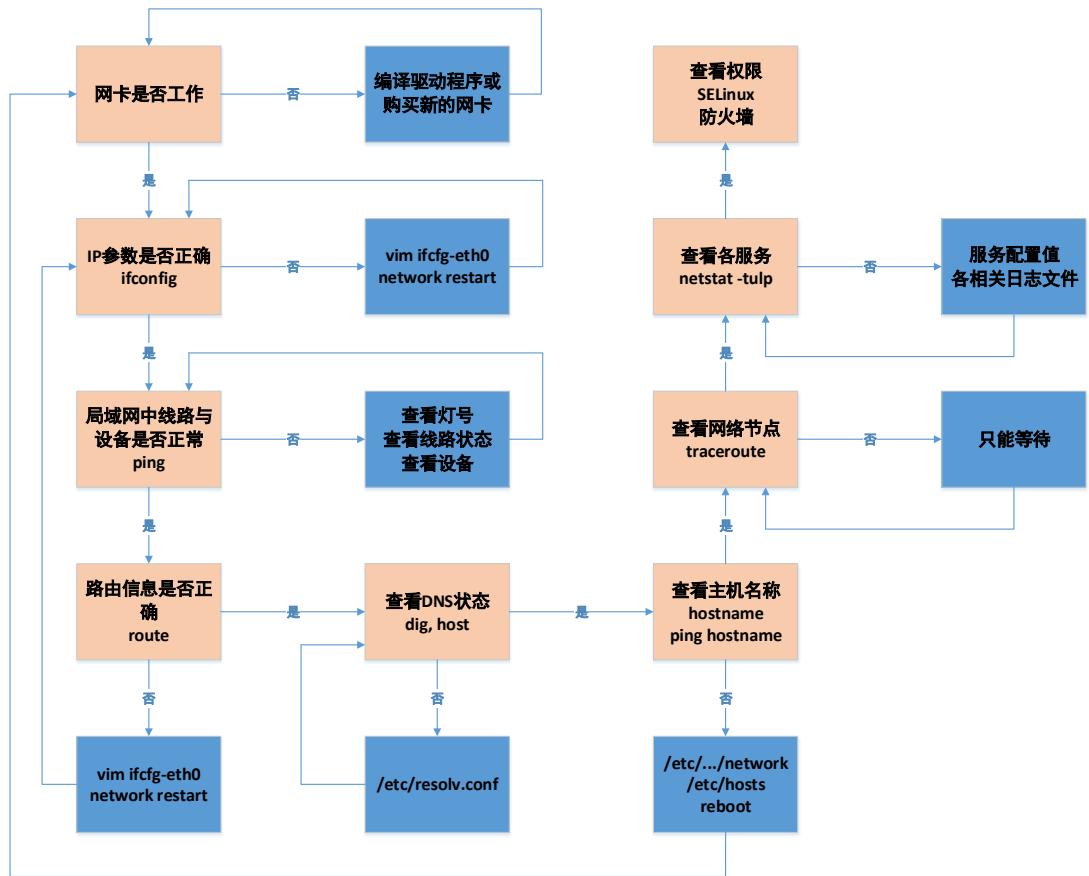


图 6-1 网络问题解决流程图

Chapter 7. 网络安全与主机基本防护：限制端口、网络升级与 SELinux

7.1. 网络数据包连接进入主机的流程

1、系统操作的基本概念非常重要

7.1.1. 数据包进入主机的流程

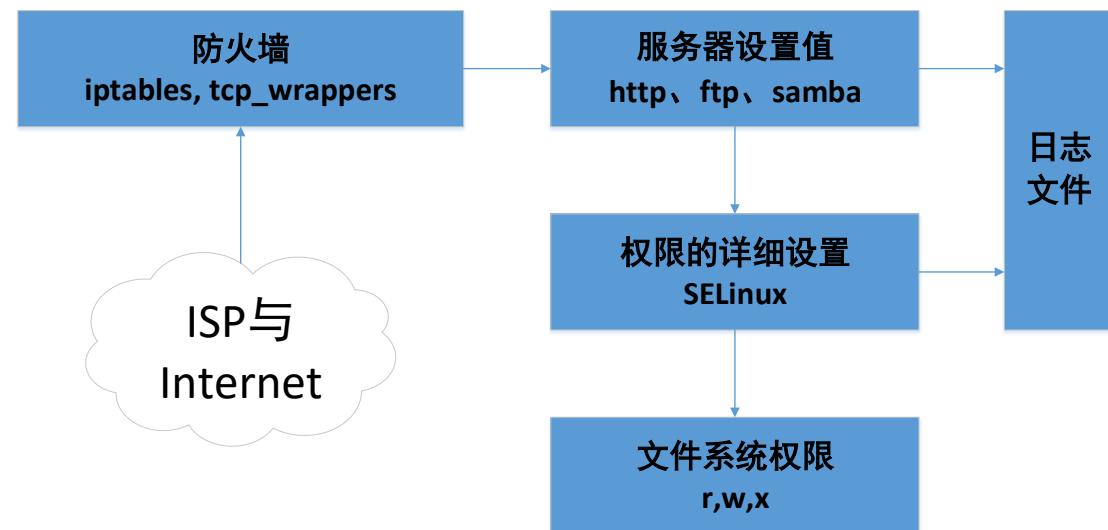


图 7-1 网络数据包进入主机的流程

1、经过防火墙的分析

- Linux 系统有内建的防火墙机制，因此你的连接能不能成功首先要通过防火墙才行
- 默认的 Linux 防火墙有两个机制，这两个机制都是独立存在的，因此默认就有两层防火墙
 - 第一层是数据包过滤式的 Net Filter 防火墙
 - 第二层则是通过软件管理的 TCP Wrappers 防火墙
- 数据包过滤防火墙：IP Filtering 或 NetFilter
 - 要进入 Linux 本机的数据包都会首先通过 Linux 内核的预置防火墙，即 Net Filter
 - 简单地说，就是 iptables 这个软件所提供的防火墙功能
 - 它主要针对 TCP/IP 的数据包头来进行过滤的机制，主要分析的是 OSI 的第二、三、四层，主要控制的是 MAC、IP、ICMP、TCP 与 UDP 的端口与状态(SYN,ACK 等)等
- 第二层防火墙：TCP Wrappers
 - 通过 Net Filter 之后，网络数据包会开始接受 Super Daemons 及 TCP Wrappers 的检验
 - 说穿了就是/etc/hosts.allow 与/etc/hosts.deny 的配置文件功能
 - 这个功能也是针对 TCP 的 Header 进行再次分析，同样可以设置一些机制来过滤某些 IP 或 Port，好让来源端的数据包被丢弃或通过检验
- 通过防火墙机制，我们可以将大部分来自因特网的垃圾连接丢弃，只允许

自己开放的服务连接进入本机，从而达到最基础的安全防护

2、服务的基本功能

- 想要设置某些目录可以进入，某些目录无法进入，需要通过权限以及如武器软件提供的相关功能来实现
- 例如，可以在 `httpd.conf` 这个配置文件之中规范某些 IP 来源不能使用 `httpd` 这个服务来取得主机的数据，那么即使该 IP 通过前面两层的过滤，它依旧无法取得主机的资源

3、SELinux 对网络服务的详细权限控制

- SELinux 可以针对网络服务的权限来设置一些规则(Policy)，让程序能够拥有的功能有限，因此当用户的文件权限设置错误，以及程序有问题时，即使使用 `root` 权限，该程序能够执行的操作也是被限制的

4、使用主机的文件系统资源

- 网络数据包最终是要向主机获取文件系统数据的
- 例如你要用 `httpd` 这个程序来取得系统的文件数据，但 `httpd` 默认是由一个系统账号名称为 `httpd` 来启动的，因此你的网页数据的权限就是要让 `httpd` 这个程序可以读取才行

7.1.2. 常见的攻击手法与相关保护

1、取得账户信息后猜密码，因此良好的密码设置习惯是很重要的

- 预防措施
 - 减少信息的曝光机会
 - 建立较严格的密码设置规则
 - 完善的权限设置

2、利用系统的程序漏洞主动攻击

- 只要获取攻击程序就可以进行攻击了，而且由攻击开始到取得你的 `root` 权限(不需要密码)，不超过两分钟就能够入侵成功
- 预防措施：
 - 关闭不需要的服务
 - 随时保持更新
 - 关闭不需要的软件功能

3、利用社会工程学欺骗

- 预防措施
 - 追踪互动者
 - 不要随意透露账号、密码等信息

4、利用程序功能的"被动"攻击

- 预防措施
 - 随时更新主机上的所有软件
 - 较小化软件的功能
 - 不要连接到不明的主机

5、蠕虫或木马的 Rootkit

- RootKit 是指可以取得 `root` 权限的一群工具组(Kit)
- RootKit 主要也是通过主机的程序漏洞进行攻击
- RootKit 不好追踪，很多时候它会主动修改系统查看的命令，包括 `ls`、`top`、`netstat`、`ps`、`who`、`w`、`last`、`find` 等，让你看不到某些有问题的程序

➤ 预防措施

- 不要随意安装不明来源的文件或者是不明网站的文件数据
- 不要让系统有太多危险的命令
- 可以定时以 RKHunter 之类的软件来追查

6、DDoS 攻击法(Distributed Denial of Service)

➤ 分布式拒绝服务供给，就是通过分撒在各地的僵尸计算机进行攻击，让你的系统所提供的服务被阻断而无法顺利地为其他用户提供服务

➤ 例如

- 当主机接收了一个带有 SYN 的 TCP 数据包，就会启动对方要求的 Port 来等待连接，并发送回应数据包(带有 SYN/ACK 标志的 TCP 数据包)，并等待 Client 的再次回应
- 如果 Client 端在发送出 SYN 的数据包后，却将来自 Server 端的确认数据包丢弃，那么 Server 就会一直空等，而且 Client 端可以通过软件功能，在短时间内持续发送这样的 SYN 数据包，那你的 Server 就会持续不断地发送确认数据包，并且开启大量的 Port 在空等，等到全部主机的 Port 启用完毕，那么系统就"挂"了

➤ DDoS 不是侵入你的系统，而是要让你的系统无法正常提供服务

7、其他

- 设置规则完善的防火墙
- 核心功能
- 日志文件与系统监控

8、小结

- 建立完善的登录密码规则限制
- 完善的主机权限设置
- 设置自动升级与修补软件漏洞以及移除危险软件
- 在每项系统服务的设置当中，强化安全设置的项目
- 利用 iptables、TCP Wrappers 强化网络防火墙
- 利用主机监控软件(MRTG 与 logwatch)来分析主机状况与日志文件

7.1.3. 主机能执行的保护操作：软件更新、减少网络服务、启动 SELinux

1、软件更新的重要性

➤ 假设你需要对全世界开放 WWW

- 那么就需要执行提供 WWW 服务的 httpd 程序，并且防火墙需要打开 Port80 让全世界都可以连接到你的 Port80，这样才是一台合格的服务器
- 如果 httpd 程序有安全方面的问题，那么防火墙是没有作用的，因为防火墙原本就要开放 Port80
- 这时需要做的就是软件持续更新到最新，**自由软件**有这样的好处
- 当你的程序有问题时，开发人员会在最短时间内提供修补程序(Patch)，并将该程序代码补充到软件更新数据库中，让一般用户可以直接通过网络来自动更新
- 因此要克服服务器软件的问题，更新系统软件就可以了

➤ 一个企业版本的 Linux Distributions 很重要，对于服务器来说，稳定与安全比什么都重要

2、认识系统服务的重要性

- 关闭端口的方式是关闭网络服务，减少网络服务可以避免很多不必要的麻烦

3、权限与 SELinux 的辅助

- ACL 可以针对单一账号或单一组进行特定的权限设置
- 通过 SELinux 来避免用户乱用系统，乱设置权限
 - SELinux 可以在程序与文件之间再加入一个权限控制，因此，即使程序与文件的权限符，但如果程序与文件的 SELinux 类型(Type)不吻合，程序也无法读取该文件
- 另外，CentOS 也针对了某些常用的网络服务制定了许多文件使用规则(Rule)，如果这些规则没有启用，那么即使权限、SELinux 类型都对了，该网络服务的功能还是无法顺利启动

7.2. 网络自动升级软件

1、除了未来架设防火墙之外，最重要的 Linux 日常管理工作莫过于软件的升级了

7.2.1. 如何进行软件升级

1、rpm：目前最常见于 Linux Distribution 当中的软件管理方式，包括 CentOS/Fedora/SuSE/Red Hat/Mandriva 等

2、tarball：

- 软件直接在自己的机器上编译，效率会比较好，但是升级的时候比较麻烦，需要重新下载新的源代码并且重新编译一次

3、dpkg：是 debian 这个 Distribution 所使用的软件管理方式，与 rpm 类似，都是通过预先编译的处理，可以让 End User 来直接使用，升级与安装

4、在 Linux 最常见的软件安装方式，即 rpm/tarball/dpkg，tarball 由于取得的是源代码，所以要用 tarball 来执行在线自动更新时不太可能的，所以仅能用 rpm 或 dpkg 这两种软件管理方式来进行在线更新了

5、由于各个 Distributions 在管理系统上都有自己独特的想法，所以在分析 rpm 或 dpkg 软件与方式上面也有所不同

- **yum**：CentOS 与 Fedora 所常用的自动升级机制，通过 FTP 或 WWW 来进行在线升级以及在线直接安装软件
- **apt**：最早由 debian 这个 Distribution 所发展，现在 B2D 也就是使用 apt，同时由于 apt 的可移植性，所以只要 rpm 可以使用 apt 来管理的话，就可以自行建立 apt 服务器来提供其他用户进行在线安装与升级
- **you(Yast Online Update)**：是由 SuSE 自行开发出来的在线安装升级方式，经过注册取得一组账号、密码后，就能够使用 you 的机制来进行在线升级
- **urpmi**：Mandriva 所提供的在线升级机制

7.2.2. CentOS 的 yum 软件更新、镜像站点使用的原理

1、yum 的基本原理是：CentOS 可以在 yum 服务器上面下载官网给出的 rpm 表头列表数据，该数据除了记载每个 rpm 软件的相依性之外，也说明了 rpm 文件所放置的容器(Repository)所在，因此通过分析这些数据，CentOS 就能够直接使用 yum 去下载与安装所需要的软件了

2、详细流程如下

- 先由配置文件判断 yum Server 所在的 IP 地址
- 连接到 yum Server 后，先下载新的 rpm 文件的表头数据
- 分析比较用户所欲安装/升级的文件，并提供用户确认
- 下载用户选择的文件到系统中的 /var/cache/yum，并进行实际安装

3、由于所下载的清单当中已经含有所有官方网站所给出的 rpm 文件的表头相依属性的关系，所以如果你想要安装的软件包包含某些尚未安装的相依软件时，yum 会顺便帮你下载所需要的其他软件，预安装后，再安装你所实际需要的软件

4、如果全世界使用 CentOS 的朋友通通连接到同一台 yum 服务器中去下载所需要的 rpm 文件，那带宽不是很容易就堵塞了吗

- 同构镜像站点来解决
- CentOS 在世界各地都有镜像站点，这些镜像站点会将官网的 yum 服务器的数据复制一份，同时在镜像站点上面也提供同样的 yum 功能，因此，你可以在任何一台 yum 服务器的镜像站点上面下载与安装软件

7.2.3. yum 的功能：安装软件组、全系统更新

1、Yum 不仅能够提供在线自动升级，它还可以用于查询，软件组的安装，整体版本的升级等

2、<yum>

- yum [option] [查询的工作项目] [相关参数]
- -y：当 yum 询问用户的意愿时，主动回答 yes 而不需要由键盘输入
- install：指定安装的软件名称，所以后面需要接软件名称
- update：进行整体的升级行为，当然也可以接某个软件，仅升级一个软件
- remove：删除某个软件，后面需要接软件名称
- search：搜寻某个软件或者是关键字
- list：列出目前 yum 所管理的所有软件名称与版本，有点类似 rpm -qa
- info：同上，类似于 rpm -qai
- clean：下载的文件被放到 /var/cache/yum，使用 clean 将它移除，可清除的项目有 packages|headers|metadata|cache|all 等
- grouplist：列出所有可使用的软件组，例如 Development Tools 之类
- groupinfo：后面接 group_name，则可了解该 group 内含的所有软件名
- groupinstall：可以安装一整组的软件组，更常与 --installroot=/some/path 共享来安装新系统
- groupremove：删除某个软件组

3、在默认情况下，yum 下载的数据除了每个容器的表头清单之外，所有下载的 rpm 文件都会在安装完毕之后予以删除，这样系统就不会有容量被下载的数据塞爆的情况了

4、yum 的配置文件 /etc/yum.conf

5、全系统更新

- yum update

7.2.4. 挑选特定的镜像站点：修改 yum 配置文件与清除 yum 缓存

1、CentOS 的镜像站点可能会选错，此时可以手动修改一下 yum 配置文件

- 连接到 CentOS 镜像站点的网址后，会发现里面有一堆链接，这些链接就

是这个 yum 服务器所提供的容器，最好认的容器就是 os(系统默认的软件)与 updates(软件升级版本)

➤ 配置文件参数解释/etc/yum.repos.d/CentOS-Base.repo

- [base]: 代表容器的名字，中括号一定要存在，里面的名称任意取，但是不能有两个相同的容器名称，否则 yum 会不知道该到哪里去找容器相关软件列表文件
- name: 只是说明一下容器的意义，重要性不高
- mirrorlist=: 列出这个容器可以使用的镜像站点，如果不使用，可以注释掉，如果要直接设置镜像站点，那么这个必须注释掉
- baseurl=: 后接容器的实际网址，mirrorlist 是由 yum 程序自行去获取镜像站点，baseurl 则是固定的容器网址
- enable=1: 启动这个容器，不启动则为 0
- gpgcheck=1: 指是否需要查阅 rpm 文件内的数字签名
- gpgkey=: 就是数字签名公钥文件所在的位置，使用默认值即可

2、<未完成>

7.3. 限制连接端口 (Port)

7.3.1. 什么是 Port

1、当你启动一个网络服务时，这个服务会依据 TCP\IP 的相关通信协议启动一个端口进行监听，那就是 TCP/UDP 数据包的 Port 了

2、网络连接是双向的，服务器端需要启动一个监听的端口，客户端需要随机启动一个端口来接受响应的数据才行

3、Port 的相关知识

- 服务器端启动的监听端口所对应的服务是固定的
- 客户端启动程序时，随机启动一个大于 1024 以上的端口
- 一台服务器可以同时提供多种服务
- 共有 65535 个 Port
 - TCP/UDP 报头数据中可以知道 Port 占用 16 个位，因此一般主机会有 65535 个 Port，而这些 Port 又分为两个部分，以 Port1024 分开
 - 只有 root 才能启动保留的 Port: 在小于 1024 端口，都是需要以 root 身份才能启动的，这些 port 主要用于一些常见的通信服务，在 Linux 系统下，常见的协议与 Port 的对应记录在/etc/services
- 是否需要三次握手：建立可靠的连接服务需要 TCP 协议，非面向连接的服务，例如 DNS 与视频系统，只需要 UDP 协议即可
- 通信协议可以启动在非正规的 Port
 - 例如 WWW 默认启动在 80 端口
 - 但是 WWW 也能在其他端口启动，但是这样一来，客户要连接到该 WWW 主机时，需要在浏览器的地方额外指定你所启用的非正规的端口才行
 - 这个启动非正规的端口功能，常常被用在一些所谓的地下网站
 - 另外某些软件默认启动在大于 1024 以上的端口，如 MySQL 数据库软件启动在 3306 端口
- 所谓的 Port 安全性
 - 没有所谓的 Port 安全性，因为 Port 的启用是服务软件所造成的，因此

真正影响网络安全的不是 Port 而是启动 Port 的那个软件

- 对安全真正有危害的是某些不安全的服务而不是开了哪些 Port

7.3.2. 端口的查看: netstat、nmap

1、常用来查看 Port 的程序有两个

- **netstat:** 在本机上面以自己的程序监测自己的 Port
- **nmap:** 通过网络的监测软件辅助, 可检测非本机上的其他网络主机, 但有为违法之虞, 因为 nmap 功能太强大了, 很多 Cracker 会直接以它来探测别人的主机

2、<netstat>

- 在作为服务器的 Linux 系统中, 开启的网络服务越少越好, 因为较少的服务容易排错(Debug)与了解安全漏洞, 并可避免不必要的入侵管道
- 列出正在监听的网络服务: `netstat -tunl`
- 列出已连接的网络连接状态: `netstat -tun`
- 删除已建立或在监听当中的连接: `netstat -tunp`
 - 如果想要将已建立或者是正在监听当中的网络服务关闭的话, 最简单的就是找出该连接的 PID, 然后 kill 掉即可

3、<nmap>

- Network exploration tool and security / port scanner
- 它是被系统管理员用来管理系统安全性检查的工具
- `nmap [扫描类型] [扫描参数] [hosts 地址与范围]`
- 以下扫描类型
 - `-sT`: 扫描 TCP 数据包已建立的连接
 - `-sS`: 扫描 TCP 数据包带有 SYN 卷标的数据
 - `-sP`: 以 ping 的方式进行扫描
 - `-sU`: 以 UDP 的数据包格式进行扫描
 - `-sO`: 以 IP 的协议进行主机的扫描
- 以下扫描参数
 - `-PT`: 使用 TCP 里头的 ping 的方式进行扫描
 - `-PI`: 使用实际的 ping 来进行扫描
 - `-p`: 这个是 port range, 例如 1024-、80-1023 等
- hosts 地址与范围
 - `192.168.1.100`: 直接写入 HOST IP 而已, 仅检查一台
 - `192.168.1.0/24`: 为 C class 的形态
 - `192.168.*.*`: 变为 B Class 的形态, 扫描范围变广
 - `192.168.1.0-50,60-100,103,200`: 变种的主机范围
- 例子:
 - `nmap localhost` <== 使用默认参数扫描本机所启用的 port, 只扫描 TCP
 - `nmap -sTU localhost` <== 扫描 TCP/UDP 这两个通信协议

7.3.3. 端口与服务的启动/关闭及开机时状态设定

1、Port 是在执行某些软件后被软件激活的, 所以要关闭某些 port, 可直接将某个程序关闭就好了

- 关闭的方法可使用 kill, 但是 kill 这个命令具有强制关闭某些程序的功能

- 弱想正常关闭该程序，可以利用系统给我们的 script 即可

2、stand alone 与 super daemon

- stand alone

- 直接执行该服务的可执行文件，让该执行文件直接加载到内存当中运行，用这种方式启动的优点是可以让该服务有较快的响应速度
- 一般来说，这种服务的启动 script 都会放置到/etc/init.d/这个目录下面
- 通常可以用/etc/init.d/sshd restart 来重启服务
- 在 CentOS7 下，利用 systemctl restart sshd.service

- super daemon

- 用一个超级服务作为总管来统一管理某些特殊的服务
- CentOS 6.x 里面使用的是 xinetd 这个 super daemon
- 这种方式启动的网络服务虽然在响应速度上会比较慢，不过，通过 super daemon 额外提供一些管理，例如控制何时启动，何时进行连接、哪个 IP 可以连进来、是否允许同时连接等
- 通常个别服务的配置文件放置在/etc/xinetd.d/当中，设置完毕后用 /etc/init.d/xinetd restart 来重新启动
- **CentOS7 没有上述"/etc/init.d/xinetd"脚本，由于 systemd 代替了 SysV 的 init**

3、默认启动的服务

- 设置开启启动：chkconfig foo on(systemctl enable foo.service)
- 设置开机不启动：chkconfig foo off(systemctl disable foo.service)
- 常见的必须要存在的系统服务

- acpid
- atd
- crond
- haldaemon
- iptables
- network
- postfix
- rsyslog
- sshd
- xinetd
- 红色标记的为在 CentOS7 下查找不到的服务

7.3.4. 安全性考虑--关闭网络服务端口

1、利用 netstat -tunlp 查询后，写一个脚本来关闭即可

- systemctl disable *.service
- systemctl stop *.service

7.4. SELinux 管理原则

1、SELinux 使用的所谓的委任式访问控制(Mandatory Access Control, MAC)，它可以针对特定的程序与特定的文件资源来进行权限的管理

- 也就是说，即使你是 root，那么在使用不同程序时，你所能取得的权限不

- 一定是 root，需要视当时该程序的设置而定
- 如此一来，针对控制的主体就变成了程序而不是用户了
 - 因此，即使你的程序使用 root 的身份合法启动，如果这个程序被攻击而被取得操作权，那改程序能做的事情还是有限的，因为被 SELinux 限制住了能执行的操作

7.4.1. SELinux 的工作模式

1、SELinux 是通过 MAC 的方式来管理程序，它控制的主体是程序，而 **目标则是改程序能否读取的文件资源**

2、概念

- **主体(Subject)**: SELinux 主要管理的就是程序，因此你可以将"主体"跟本章谈到的 Process 划上等号
- **目标(Object)**: 主体程序访问的目标资源一般就是文件系统，因此这个目标项目可以跟文件系统划上等号
- **策略(Policy)**: 由于程序与文件数量庞大，因此 SELinux 会依据某些服务来制定基本的访问安全性策略。这些策略内还会有详细的规则(Rule)来指定不同的服务开放某些资源的访问与否
 - 目前 CentOS 提供 targetd 和 mls 两个主要的策略，一般来说，使用默认的 targeted 策略即可
 - targeted: 针对网络服务限制比较多，针对本机限制少，是默认的策略
 - mls: 完整的 SELinux 限制，限制方面较为严格
- **安全性环境(Security Context)**: 目标与主体的安全性环境必须一致才能够顺利访问目标
 - 这个安全性环境有点类似于文件系统的 rwx

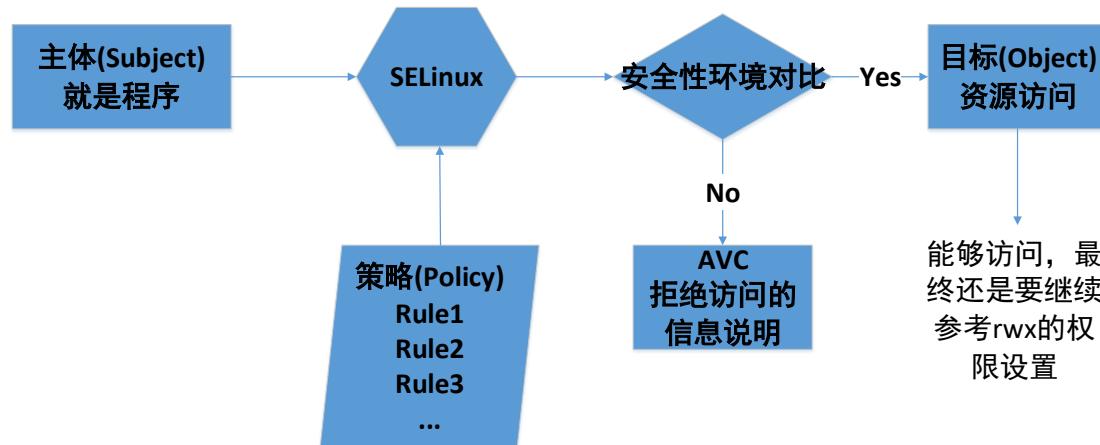


图 7-2 SELinux 工作的各组件的相关性

3、根据图 7-2

- 主体程序必须通过 SELinux 策略内的规则放行后，才可以与目标资源进行安全性对比，若比对失败则无法访问目标
- 若对比成功则可以开始访问目标
- 问题是，最终能否访问目标还是与文件系统的 rwx 权限设置有关

4、安全性环境

- 安全性环境比较麻烦，因为可能要自行配置文件的安全性环境，可以将安全性环境看成 SELinux 内必备的 rwx，这样比较好理解

- 安全性环境存在于主体程序中与目标文件资源中
 - 程序在内存内，所以安全性环境可以存入是没问题的
 - **文件的安全性环境记录是放置到文件的 inode 内的**
 - 主体程序想要读取目标文件资源时，同样需要读取 inode 就可以比对安全性环境以及 rwx 等权限值是否正确
- 利用 ls -Z 查看安全性环境
- 安全性环境主要分为三个字段
 - **身份识别(Identify)**: 相当于账号方面的身份识别，主要的身份识别有以下 3 种常见的类型
 - root: 表示 root 的账号身份
 - **system_u: 表示系统程序方面的识别 m，通常就是程序**
 - user_u: 代表的是一般用户账号相关的身份
 - **角色(Role)**: 通过角色字段，我们可以知道这个数据是代表程序、文件资源还是用户
 - **object_r: 代表的是文件或目录等文件资源，这是最常见的**
 - **system_r: 代表的就是程序，一般用户也被指定称为 system_r**
 - **类型(Type)**: 在默认的 targeted 策略中，Identify 与 Role 字段基本上是不重要的，重要的在于 Type 字段
 - **Type: 在文件资源(Object)中称为类型(Type)**
 - **Domain: 在主体程序(Subject)中称为域(Domain)**
 - Domian 需要预 Type 搭配，该程序才能够顺利地读取文件资源

5、程序与文件 SELinux Type 字段的相关性

表格 7-1 **主体程序**在三个字段中对应的意义

| 身份识别 | 角色 | 该对应在 targeted 的意义 |
|----------|----------|------------------------|
| root | system_r | 代表供 root 账号登录时所取得的权限 |
| system_u | system_r | 由于为系统账号，因此是非交互式的系统运行程序 |
| user_u | system_r | 一般可登陆用户的程序 |

- 最重要的字段是类型字段，主体与目标之间是否具有可读写的权限，与程序的 Domain 及文件的 Type 有关
- 以 httpd 为例来说明
 - httpd 属于 httpd_exec_t 这个可执行的类型
 - /var/www/html 则属于 httpd_sys_content_t 这个可让 httpd 域(Domain)读取的类型



图 7-3 主体程序取得的 Domain 与目标文件资源的 Type 之间的关系

- 上图步骤解析
 - 首先，触发一个可执行的目标文件，(httpd_exec_t 这个类型的 /usr/sbin/httpd)
 - 该文件的类型会让这个文件造成的主体程序(Subject)具有 httpd 这个域，**我们的策略针对这个域制定了许多规则，其中包括这个域可以读取的目标资源类型**

- 由于 httpd domain 被设置为可以读取 httpd_sys_content_t 这个类型的目
标文件(Object)，因此将网页放置到 /var/www/html/ 目录下，就能够被
httpd 程序所读取了
- 最终能否读到正确的资料，还要看 rwx 是否符合 Linux 权限的规范
- 几个重点
 - 第一个是策略内需要制定详细的 domain/type 相关性
 - 第二个是若文件的 Type 设置错误，即使权限设置为 rwx 全开的 777，改
主体程序也无法读取到目标文件资源，但是可以避免用户将它的主目录
设置为 777 时所造成的权限困扰

7.4.2. SELinux 的启动、关闭与查看

1、目前 SELinux 支持三种模式

- enforcing：强制模式，代表 SELinux 运行中，且已经正确的开始限制
domain/type 了
- permissive：宽容模式，代表 SELinux 运作中，不过仅会有警告信息并不会
实际限制 Domain/Type 的访问，这种模式可以用来作为 SELinux 的 Debug
之用
- disabled：关闭，SELinux 并没有实际运行

2、<getenforce>

- 获取 SELinux 的运行模式

3、配置文件/etc/selinux/config

- 如果改变了策略需要重新启动
- 如果由 enforcing 或 permissive 改成 disabled 或由 disabled 改成其他两
个，也必须重新启动
- 这是因为 SELinux 是整合到内核里面去的，你只能在 SELinux 运行下切换
称为强制(Enforcing)或宽容(Permissive)模式，不能够直接关闭 SELinux
- 如果是 disabled 状态，那只能通过修改配置文件，然后重新启动的方式

4、<setenforce>

- 让 SELinux 模式在 Enforcing 与 Permissive 之间切换
- setenforce [0|1]
- 0：转成 permissive 宽容模式
- 1：转成 Enforcing 强制模式

7.4.3. SELinux Type 的修改

1、当用 cp 来复制时，SELinux 的 Type 字段会继承自目标目录

2、当用 mv 来移动时，连同 SELinux 的类型也会被移动过去

3、<chcon>

- chcon [-R] [-t type] [-u user] [-r role] 文件
- chcon [-R] --reference=范例文件 文件
- -R：连同该目录下的子目录也同时修改
- -t：后面接安全性环境的类型字段
- -u：后面接身份识别，例如 system_u
- -r：后面接角色，例如 system_r
- --reference=范例文件：拿某个文件当返利来修改后续的文件类型

4、<restorecon>

- 恢复成原有的 SELinux Type
- restorecon [-Rv] 文件或目录
- -R: 连同子目录一起修改
- -v: 将过程显示到屏幕上

5、semanage

- restorecon 怎么会知道每个目录记载的默认 SELinux Type?因为系统有记录, 记录在/etc/selinux/targeted/contexts 这个目录中, 使用文本编辑器去查阅比较麻烦, 可以通过 semanage 来查询与修改
- semanage {login|user|port|interface|fcontext|translation} -l
- semanage fcontext -{a|d|m} [-first] file_spec
- fcontext: 主要用在安全性环境方面
- -l: 查询
- -a: 增加
- -m: 修改
- -d: 删除
- 例子
 - senange fcontext -a -t public_content_t "/srv/liuye(/.*)?" <==这里是正则表达式
 - 上述命令会将记录写入 /etc/selinux/targeted/contexts/files/file_contexts.local
- semanage 的功能很多, 不过主要用到的就是 fcontext 这个项目操作而已

7.4.4. SELinux 策略内的规则布尔值修订

1、要通过 SELinux 的验证之后才能开始文件权限 rwx 的判断

2、策略查询

- 可以通过<seinfo>来查询
 - yum install setools-console
 - seinfo [-Atrub]
 - -A: 列出 SELinux 的状态、规则布尔值、身份识别、角色、类别等所有信息
 - -t: 列出 SELinux 的所有类别(type)种类
 - -r: 列出 SELinux 的所有角色(role)种类
 - -u: 列出 SELinux 的所有身份识别(user)种类
 - -b: 列出所有规则的种类(布尔值)
- 通过<seseach>查询详细规则
 - seseach [--all] [-s 主体类别] [-t 目标类型] [-b 布尔值]
 - --all: 列出该类别或布尔值的所有相关信息
 - -t: 后面还要接类别, 例如 -t httpd_t
 - -b: 后面接布尔值规则, 例如 -b httpd_enable_ftp_server
 - seseach --all -t httpd_sys_content_t

3、布尔值的查询与修改

- <getsebool>

- `getsebool [-a]` [布尔值条款]
- -a: 列出目前系统上面的所有布尔值条款设置为开启或关闭值
- <setsebool> [-P] 布尔值=[0|1]
- -P: 直接将设置值写入配置文件, 改设置数据未来会生效
- 最好记得加上-P 选项, 这样才能将设置写入配置文件

7.4.5. SELinux 日志文件记录所需的服务

1、<setroubleshoot>: 将错误信息写入/var/log/messages

- 几乎所有 SELinux 相关的程序都会以 se 开头
- 这个服务会将关于 SELinux 的错误信息与克服方法记录到 /var/log/messages 与 /var/log/setroubleshoot 中
- **这个服务需要两个软件, 分别是 setroubleshoot 与 setroubleshoot-server**
- 例子, 加入有个程序所要读取的文件的安全性环境不匹配(由于设置错误)
 - `cat /var/log/messages | grep setroubleshoot`
 - 会看到这样的东西"sealert -l <一串数字和字符>"
 - 运行这个命令, 就会看到解决方法, CentOS7 好像在最开始的地方出现

2、用 E-mail 或在命令列上面直接提供 setroubleshoot 错误信息

- 如果每次测试都要到 /var/log/messages 去分析, 那真是挺麻烦的
- 我们可以通过 E-mail 或 console 的方式来产生信息, 我们可以让 setroubleshoot 主动发送产生的信息到指定的 E-mail, 这样可以方便实时分析
- 修改 setroubleshoot 的配置文件 /etc/setroubleshoot/setroubleshoot.cfg
 - CentOS7 中的后缀是".conf"
 - 在 81 行加上
"recipients_filepath = /var/lib/setroubleshoot/email_alert_recipients", 如果有就不用加了
 - 将 147 行原本的"console = False"改为"console = True"
 - 然后"vim /var/lib/setroubleshoot/email_alert_recipients", 添加两句
 - root@localhost //这样本机的 root 才能收到信件
 - your@email.address

3、SELinux 错误克服的总结

- 网络连接要通过 SELinux 的权限判定后才能够继续 rwx 的权限对比
- 而 SELinux 的对比需要通过策略的各项规则比对后才能够进行 SELinux Type 安全性环境的比对, 这两项工作需要正确才行
 - 而修改主要通过 chcon、restorecon、setsebool 等命令来处理
 - 怎么知道如何处理?, 通过分析 /var/log/messages 内提供的 setroubleshoot 的信息
- **建议处理流程**
 - 在服务与 rwx 权限都没问题, 却无法成功使用网络服务时, 先使用 setenforce 0 设置为 permissive 模式
 - 再次使用网络服务, 如果能用, 表示 SELinux 出现问题, 往下处理
 - 分析 /var/log/messages 内的信息, 找到"sealert -l"相关信息并且执行
 - 找到开头处理方式, 并执行
 - 处理完毕重新 setenforce 1, 再次测试网络服务

7.5. 被攻击后的主机修复工作

7.5.1. 网管人员应具备的技能

1 一台主机最常发生问题的状况，都是由内部的网络误用所产生的

2、需要的技巧

- 了解什么是需要保护的内容
- 预防黑客入侵
- 主机环境安全化
- 防火墙规则制定
- 实时维护主机
- 良好的教育训练课程
- 完善的备份计划

7.5.2. 主机收到攻击后恢复的工作流程

1、万一主机被入侵了，最好的方法还是"重新安装 Linux"，流程如下

2、立即拔除网线

3、分析日志文件信息，查找可能的入侵途径

- 分析日志文件：低级的 Cracker 通常仅是利用工具软件来入侵系统，可以通过分析一些主要的日志文件来找出对方 IP 以及可能有问题的漏洞
 - /var/log/messages
 - /var/log/secure
 - last 命令
 - 检查主机开放的服务
 - 查询 Internet 上面的安全通报
- 3、重要数据备份
- 重要的数据是指非 Linux 系统上原有的数据
 - /etc/passwd、/etc/shadow、WWW 网页的数据、/home 里面的用户重要文件
 - 至于/etc/*、/usr、/var 等目录下的数据就不见得要备份了
- 4、重新安装
- 5、软件的漏洞修补
- 6、关闭或删除不需要的服务
- 7、数据恢复与恢复服务设置
- 8、连接上 Internet

Chapter 8. 路由的概念与路由器设置

8.1. 路由

8.1.1. 路由表产生的类型

1、每一台主机都有自己的路由表，必须要通过自己的路由表将主机的数据包转发到下一个路由器，发送出去后，该数据包就要通过下一个路由器的路由表来传送了，此时与你自己的主机的路由表就无关了

2、通过 route -n 来查看路由表

3、在 Linux 系统下的路由表是由小网络排列到大网络的

4、路由表设计的依据

- 依据网络接口产生的 IP 而存在的路由
 - 主机上面有几个网络接口存在时，该网络接口就会存在一个路由
- 手动或默认路由(Default Route)
 - 默认路由(0.0.0.0/0)就是额外路由
 - 你所规划的路由必须要是你的设备(如 eth0)或 IP 可以直接沟通的情况
- 动态路由
 - 通过路由器与路由器之间的协商以实现动态路由的环境
 - 需要额外的软件支持，例如 zebra 或 CentOS 上的 Quagga

5、事实上，Linux 的路由规则都是通过内核来实现的，所以这些路由表的规则都是在内核功能，也就是运行在内存中

8.1.2. 一个网卡绑定多个 IP: IP Alias 的测试用途

1、一个网卡可以具有多个 IP(例如 eth0:0)，具有多个 IP 的功能就被称为 IP Alias

- eth0:0 的设备可以通过 ifconfig 或 ip 这两个命令来实现

2、IP Alias 的用途

- 测试用
 - 现在 IP 路由器的设置通常是使用 WWW 接口来提供的，这个 IP 路由器通常会给予一个私有 IP，即 192.168.0.1 让用户开启 WWW 接口的浏览器
 - 那你要如何连接上这台 IP 路由器？在不改变网络环境的情况下，可以直接利用以下方式

```
ifconfig [device] [IP] netmask [netmask ip] [up|down]
ifconfig eth0:0 192.168.0.100 netmask 255.255.255.0 up
```

- 这样就建立了一个虚拟的网络接口，可以立即连接上 IP 路由器了，也不会影响到原来的网络参数设置值

➤ 在一个实体网络中含有多个 IP 网络

- 不允许修改主机的网络设置，并且让大家胡同所有计算机的信息，可以让每个主机都通过 IP Alias 来设置同一个网络的 IP

➤ 既有设备无法提供更多实体网卡时

3、所有的 IP Alias 都是由实体网卡仿真的，所以当要启动 eth0:0 时，eth0 必须先被启动才行，当 eth0 停关闭后，所有 eth0:n 的模拟网卡也将同时被关闭

4、若要该 IP Alias 开机启动，可以建立一个虚拟设备的配置文件

- cd /etc/sysconfig/network-scripts

- vim ifcfg-eth0:0
 - DEVICE=eth0:0
 - ONBOOT=yes
 - BOOTPROTO=static
 - IPADDR=192.168.0.100
 - NETMASK=255.255.255.0
- 无论 ifcfg-eth0:0 内的 ONBOOT 设置值为何，只要 ifcfg-eth0 这个实体网卡的配置文件中 ONBOOT 为 yes 时，开机就会将全部的 eth0:n 都启动

8.1.3. 重复路由的问题

- 1、可不可以利用两张网卡、两个相同网络的 IP 来增加主机的网络流量？这是一个可行的方案，不过必须要通过许多的设置来完成
 - 一般来说不应该设置同一网段的不同 IP 在同一台主机上面

8.2. 路由器配置

- 1、局域网内主机可以通过广播的方式来进行网络数据包的发送，但在不同网段内的主机想要互相链接时就需要通过路由器了

8.2.1. 什么是路由器与 IP 路由器

- 1、主机想要将数据传送到不同的网段时需要通过路由器帮忙，路由器的主要功能就是转发网络数据包
 - 路由器会分析来源端数据包的 IP 包头，在包头内找出要送达的目标 IP 后，通过路由器本身的路由表(Routing Table)来将这个数据包向下一个目标(Next Hop)传送

2、路由器功能如何实现

- **硬件功能**: 如 Cisco、TP-Link、D-Link(注 2)等公司都产生硬件路由器，这些路由器内有嵌入式的操作系统，可以负责不同网段内的数据包翻译与传递等功能
- **软件功能**: 如 Linux 这个操作系统的内核就提供了数据包传递的能力
- 本章仅讨论在以太网中最简单的路由器功能：连接两个不同的网段，该功能在 Linux 个人计算机中就可以实现

3、路由表是由 Linux 的内核功能提供的，转发数据包的能力也是 Linux 内核提供的，如何查看内核是否已经启动数据包转发呢

- cat /proc/sys/net/ipv4/ip_forward <=0 代表没有启动，1 代表启动了
- 让该文件内容变为 1, echo 1 > /proc/sys/net/ipv4/ip_forward，不过这个设置在下次重启后会失效
- 可以修改/etc/sysctl.conf 来实现开机启动数据包转发的功能
 - net.ipv4.ip_forward = 1(将 0 值改为 1 即可，没有就添加这句)
 - 然后 sysctl -p 使其立即生效

4、静态路由

- 直接以类似 route 这个命令来直接设置路由表到内核功能当中，设置值值要与网段环境相符即可
- 当网段有变化时，路由器就要重新设置

5、动态路由

- 通过类似 Quagga 或 zebra 软件的功能，这些软件可以安装在 Linux 路由器上
- 这些软件可以动态监测网络的变化，并直接修改 Linux 内核的路由表信息，无须手动以 Route 来修改路由表信息

5、NAT(Network Address Translation, 网络地址转换)

- IP 路由器就是最简单的 NAT 服务器
- NAT 可以实现 IP 共享的功能，而 NAT 本身就是一个路由器，只是 NAT 比路由器多了一个 IP 转换的功能
- NAT 与路由器差别如下
 - 一般来说，路由器会有两个网络接口，通过路由器本身的 IP 网段转发功能能让两个网段可以相互沟通网络数据包，如果两个接口，一个是公共 IP(Public IP)，但一个是私有 IP(Private IP)，由于私有 IP 不能直接与公共 IP 沟通其路由信息，此时需要额外的"IP 转换"功能
 - Linux 的 NAT 服务器可以通过修改数据包的 IP 包头数据的来源或目标 IP，来让私有 IP 的数据包可以转成 NAT 服务器的公共 IP，直接连上 Internet
 - 所以，当路由器两端的网络分别是 Public IP 与 Private IP 时才需要 NAT 的功能

8.2.2. 何时需要路由器

1、一般来说，小于 10 台的小型企业是无需路由器的，只需要利用 Hub/Switch 连接各台计算机，然后通过单一线路连接到 Internet 即可

2、实际线路的布线以及效能的考虑

- 在一栋大楼所有不同层连接所有计算机可能有难度
- 通过在每个楼层架设一台路由器，并将每个楼层路由器相连接
- 如果不想配置路由器，而是直接以以太网线连接各楼层的 Hub/Switch 时，由于同一网段的数据是通过广播来传递的，会造成网络效能的问题，因此架设路由器是理想选择

3、部门独立于保护数据的考虑

- 只要实际线路是连接在一起的，那么当数据通过广播时，就可以通过类似 tcpdump 的命令来监听数据包，并窃取
- 如果部门间的数据需要独立，或者是某些重要的数据必须要在公司内部予以保护时，可以将重要的计算机放到独立的实际网段，并额外加设防火墙，路由器等连接上公司内部的网络

8.2.3. 静态路由的路由器

1、<未完成>：详见 P234，一个实例

2、**连接两个 Private IP 的 Linux Router 很简单，将两块网卡设置两个 IP，并且启动内核的数据包转发功能，立即就架设完毕了**

3、平台：Windows

- VM Linux 虚拟机 1，作为 Linux Router
- /etc/sysconfig/network-scripts/ifcfg-eno16777736
DEVICE=eno16777736 <==必须与文件名 ifcfg-后面那串字符相同
HWADDR="00:0c:29:a7:5d:b6" <==虚拟 MAC 地址

```

NM_CONTROLLED="no"
ONBOOT="yes"    <==启动
BOOTPROTO=none   <==手动配置 IP, 不用 dhcp
IPADDR=192.168.136.166 <==在 192.168.136.0/24 任意设置一个 IP
NETMASK=255.255.255.0 <==该网段的子网掩码
GATEWAY=192.168.136.2 <==这个不知道是哪个设备的 IP, 可能是 VM 平
台下的一个虚拟的路由器
● 由于没有两张网卡, 于是通过 IP Alias 来构建第二张网卡
● /etc/sysconfig/network-scripts/ifcfg-eno16777736:0
DEVICE=eno16777736:0
ONBOOT=yes
BOOTPROTO=static
IPADDR=192.168.200.254
NETMASK=255.255.255.0
# GATEWAY=192.168.136.130 <==这里不能设置哦
● echo "1" > /proc/sys/net/ipv4/ip_forward <==改完立即生效的, 重启后
失效(如果与原值不同的话)
● 或者可以修改配置文件 vim /etc/sysctl.conf
■ net.ipv4.ip_forward = 1
■ 修改完成后, 运行 sysctl -p, 重新加载一下
● iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
■ 在 iptables 规则较前面的位置加上这句, 否则其他主机无法 ping 通该
主机
➤ VM Linux 虚拟机 2, 作为 192.168.200.0/24 网段的另一个主机
● /etc/sysconfig/network-scripts/ifcfg-eno16777736
DEVICE="eno16777736"
HWADDR="00:0c:29:04:28:21"
NM_CONTROLLED="no"
ONBOOT="yes"
BOOTPROTO=none
IPADDR=192.168.200.10 <==在 192.168.200.0/24 任意设置一个 IP
NETMASK=255.255.255.0
GATEWAY=192.168.200.254 <==将 Linux 虚拟机 1 作为路由地址

```

4、碰到的问题

- 若关闭内核的数据包转发功能, VM Linux 虚拟机 2 无法正常上网, 但是能 ping 通 192.168.136.166
- 当 VMWare 打开 DHCP 功能, 即 Windows 的其中一个 ip 为 192.168.136.*
- Windows 能 ping 通 192.168.136.166, 但是 ping 不通 192.168.136.2(VM Linux 虚拟机 1 的网关), 也 ping 不通 192.168.200.254(VM Linux 虚拟机 1 的另一个网卡接口)
- VM Linux 虚拟机 2(192.168.200.101)可以 ping 通 windows
- Windows 能 ssh 连上 192.168.136.166, 连不上 192.168.200.254 和 192.168.200.101
- Windows 不能用 vnc 连上 192.168.136.166、192.168.200.254、

192.168.200.101

- 当 VMWare 打开 DHCP 功能，由 Linux 虚拟机 1 提供 DHCP 功能时，Windows 的其中一个 IP 为 192.168.200.*
 - 此时 Linux 虚拟机 1 的数据包转发功能已经打开
 - Windows 通过 ping 可连通 192.168.200.254/101，连不通 192.168.136.166
 - Windows 通过 ssh 可连通 192.168.200.254，连不通 192.168.136.166
 - Windows 通过 VNC 可连通 192.168.200.254，连不通 192.168.136.166
 - 结论：???

8. 2. 4. 动态路由器架设

1、常见的动态路由协议有 RIPv1、RIPv2、OSPF、BGP 等

2、上一小节架设了一台 Linux Router1

- 整个局域网的路由器 IP 为 192.168.136.2
- Linux Router1 IP 为 192.168.136.166
- Linux Router 1 的子网 IP 为 192.168.166.1
- 子网 192.168.166.0/24 下的主机为 192.168.166.2

3、另一台 Linux Router2 的架设

- /etc/network/interfaces <==Ubuntu 上面网卡配置文件
- auto ens33 <==开机自动启动
- iface ens33 inet static <==手动配置
- address 192.168.136.66
- netmask 255.255.255.0
- gateway 192.168.136.2

```
auto ens33:0
iface ens33:0 inet static
    address 192.168.168.1
    netmask 255.255.255.0
```

4、在两台 Router 上面设置 zebra

8. 3. 特殊状况—路由器两边界面是同一个 IP 网段：ARP Proxy

1、<未完成>

Chapter 9. 防火墙与 NAT 服务器

1、防火墙就是通过定义一些有顺序的规则，并管理进入到网络内的主机数据包的一种机制，更广义地说，只要能够分析与过滤进出我们管理的网络的数据包的数据，就可以称为防火墙

2、防火墙又可以分为硬件防火墙与本机的软件防火墙

- 硬件防火墙是由厂商设计好的主机硬件，这台硬件防火墙内的操作系统主要以供数据包数据的过滤机制为主，将其他不必要的功能拿掉
- 软件防火墙本身就是保护系统网络安全的一套软件(或称为机制)，例如 Netfilter 与 TCP Wrappers 都可以称为软件防火墙

9.1.1. 关于本章的一些提醒事项

9.1.2. 为何需要防火墙

1、数据包进入本机时，会通过防火墙、服务器软件程序、SELinux 与文件系统等，如果你的系统已经采取以下操作，那么实际上已经较为安全了

- 已关闭不需要且危险的服务
- 已经将整个系统的所有软件都保持在最新状态
- 权限设置妥当且定时进行备份工作
- 已经教育用户具有良好的网络、系统操作习惯

2、**防火墙最大的功能就是帮助你限制某些服务的访问来源**

- 可以限制文件传输服务(FTP)只在子域内的主机才能够使用，而不对整个 Internet 开放
- 可以限制整台 Linux 主机尽可以接受客户端的 WWW 要求，其他的服务都关闭
- 可以限制整台主机仅能主动对外连接，也就是说，若有客户端对我们主机发送主动连接的数据包状态(TCP 数据包的 SYN flag)就予以过滤

3、防火墙最重要的任务就是规划出

- 切割被信任(如子域)与不被信任(如 Internet)的网段
- 划分出可提供 Internet 的服务于必须受保护的服务
- 分析出可接受与不可接受的数据包状态

9.1.3. Linux 系统上防火墙的主要类别

1、依据防火墙管理的范围，我们可以将防火墙区分为网络型与单一主机型的管理

- 单一主机型的管理方面：主要的防火墙有数据包过滤型的 Netfilter 与依据服务软件程序作为分析的 TCP Wrappers 两种
- 若以区域型的防火墙而言，由于此类防火墙都是充当路由器角色，因此防火墙类型则主要有数据包过滤的 Netfilter 与利用代理服务器(Proxy Server)进行访问代理的方式

2、Netfilter(数据包过滤机制)

- 所谓数据包过滤，也就是分析进入主机的网络数据包，将数据包的头部数据提取出来进行分析，以决定改连接为放行或抵挡的机制

- 这种方式可以直接分析数据包头部数据，包括硬件地址(MAC)、软件地址(IP)、TCP、UDP、ICMP 等数据包的信息进行过滤分析
- 在 Linux 上，我们使用内核内建了 Netfilter 这个机制，而 Netfilter 提供了 iptables 这个软件来作为防火墙数据包过滤的命令
 - 由于 Netfilter 是内核内建的功能，因此效率非常高，非常适合于一般小型环境的设置
 - Netfilter 利用一些数据包过滤的规则设置，来定义出什么数据可以接收，什么数据需要剔除，以达到保护主机的目的

3、TCP Wrappers(程序管理)

- 另一种抵挡数据包进入的方法，是通过服务器程序的外挂(tcpd)来处置
- 与数据包过滤不同，这种机制主要是分析谁对某程序进行访问，然后通过规则去分析该服务器程序谁能够连接，谁不能连接
- **由于主要是通过分析服务器程序来管理，因此与启动的端口无关，只与程序的名称有关**

4、Proxy(代理服务器)

- 代理服务器是一种网络服务，它可以代理用户的需求，代为前往服务器取得相关的资料
- 一般 Proxy 主机通常仅开放 port80、21、20 等 WWW 与 FTP 的端口，而且通常 Proxy 就架设在路由器上面，因此可以完整地掌控局域网内的对外连接，让 LAN 变得更安全

9.1.4. 防火墙的一般网络布线示意

1、防火墙除了可以保护防火墙机制本身所在的主机之外，还能保护防火墙后面的主机，也就是说，防火墙除了可以防备本机被入侵之外，它还可以架设在路由器上面借以控制进出本地端网络的网络数据包

2、单一网络，仅有一个路由器 P256

- 防火墙可以架设在路由器上面以管理整个局域网的数据包进出
- 因此，在这类的防火墙上通常至少需要有两个接口，将可信任的内部与不可信任的 Internet 分开，因此可以分别设置两块网络接口的防火墙规则
- 优势：
 - 内网网络分开，所以安全维护在内部可以开放的权限比较大
 - 安全机制的设置可以针对 Linux 防火墙主机来维护即可
 - 对外只看到 Linux 防火墙主机，所以对于内部可以达到有效的安全防护

3、内部网络包含安全性更高的子网，需内部防火墙切开子网 P256

4、在防火墙的后面架设网络服务器主机 P257

- 所有主机都具有相同的 Public IP，即防火墙主机的 IP，当使用各种方法进攻主机，其实攻击的都是防火墙主机，除非搞定防火墙，否则很难入侵内部主机
- P257 的主机对于内部网络与 Internet 都架设了防火墙，因此内部网络发生状况，也不会对主机造成影响
- 将网络服务器独立放置在两个防火墙中间的网络，称之为非军事化隔离区(DMZ)。其目的重点就在保护服务器本身，将 Internet 与 LAN 都隔离开来，如此一来，无论服务器本身或者是 LAN 被攻陷，另一个区域都还是完好无损的

9.1.5. 防火墙的使用限制

1、Linux 的 Netfilter 机制可以做的事情

- 拒绝让 Internet 的数据包进入主机的某些端口
- 拒绝让某些来源 IP 的数据包进入
- 拒绝让带有某些特殊标志(flag)的数据包进入
 - 最常拒绝的就是带有 SYN 的主动连接的标志
- 分析硬件地址(MAC)来决定连接与否
 - 由于 MAC 地址是焊在网卡上的，利用防火墙将某些 MAC 锁住，除非更换网卡，否则换 IP 是没用的

2、Netfilter 机制无法做到的事

- 防火墙并不能有效阻挡病毒或木马程序：如果请求 WWW 服务的数据包本身就是病毒程序的一部分时，防火墙是无用的，因为防火墙本来的设置规则就是让请求 WWW 服务的数据包通过
- 防火墙对于来自内部 LAN 的攻击无能为力

9.2. TCP Wrappers

1、TCP Wrappers 通过客户端想要链接的程序文件名，然后分析客户端的 IP，看看是否需要放行

9.2.1. 哪些服务有支持

1、说穿了，TCP Wrappers 就是通过/etc/hosts.allow、/etc/hosts.deny 这两个配置文件来管理的一个类似防火墙的机制

- 并非所有软件都可以通过这两个文件来管理，只有下面的软件才能通过这两个文件来管理防火墙规则
 - 由 super daemon 所管理的服务
 - 支持 libwrap.so 模块的服务

2、由于支持 TCP Wrappers 的服务必定包含 libwrap 这一个动态函数库，因此可以使用 ldd 来查看该服务即可

- ldd \$(which rsyslogd sshd xinetd httpd)
- for name in rsyslogd sshd xinetd httpd; do echo \$name; \> ldd \$(which \$name) | grep libwrap; done

9.2.2. /etc/hosts.{allow|deny} 的设置方式

1、语法

- <service(program_name)>: <ip, domain, hostname>
- <服务(也就是程序名字)>: <IP 或 域 或 主机名>
- # ><是不存在于文件中的

2、规则

- 先以/etc/hosts.allow 进行优先比对，该规则符合就予以放行
- 再以/etc/hosts.deny 比对，规则符合就予以抵挡
- 若不在这两个文件内，亦规则都不符合，最终则予以放行

9.3. Linux 的数据包过滤软件: iptables

9.3.1. 不同 Linux 内核版本的防火墙软件

1、内核支持的防火墙是不断演变的

- Version 2.0: 使用 ipfwadm 这个防火墙机制
- Version 2.2: 使用的是 ipchains 这个防火墙机制
- Version 2.4 与 2.6: 主要使用 iptables 这个防火墙机制
- 查看内核版本 uname -r

2、CentOS7 下防火墙的关闭与启动

- systemctl stop firewalld.service
- systemctl restart firewalld.service

9.3.2. 数据包进入流程: 规则顺序的重要性

1、iptables 利用的是数据包过滤的机制，它会分析数据包的报头数据，根据报头数据与定义的规则来决定改数据包是否可以进入主机或者是被丢弃

2、根据数据包的分析资料"对比"预先定义的规则内容，若数据包数据与规则内容相同则进行动作，否则就继续下一条规则的比对。**重点在比对与分析顺序**

- 当一个网络数据包要进入到主机之前，会先经过 Netfilter 进行检查，那就是 iptables 的规则，检查通过则接受(ACCEPT)进入本机取得资源，如果检查不通过，则可能予以丢弃(DROP)

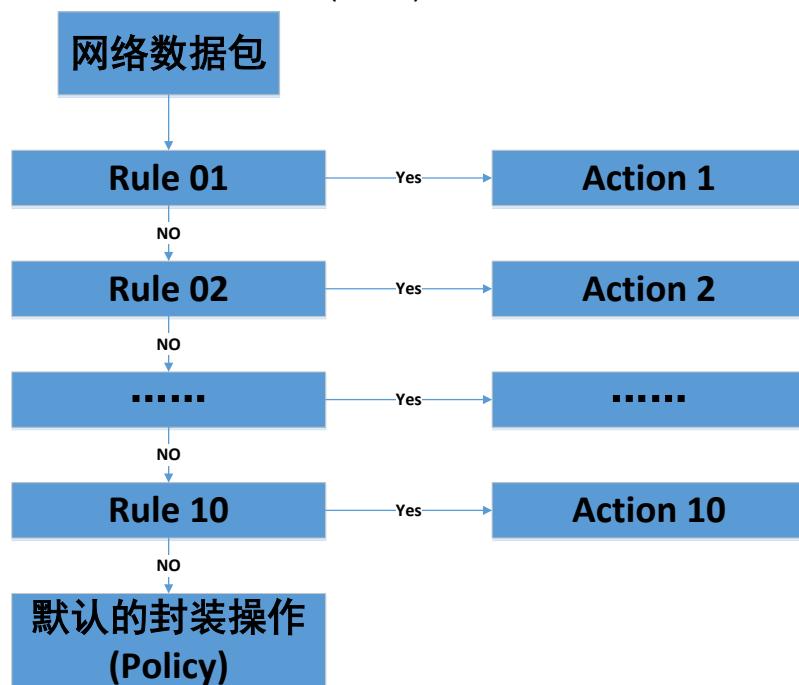


图 9-1 数据包过滤的规则操作及分析流程

- 如果比对结果符合 Rule1，此时这个网络数据包就会进行 Action1 的动作，而不会理会后续的 Rule2 Rule3 等规则了
- **当规则顺序排列错误时，就会产生很严重的错误**
 - 例如
 - Rule1: 阻挡 192.168.100.100
 - Rule2: 再让请求 WWW 服务的数据包通过

Rule3: 将所有的数据包丢弃

- 如果变为

Rule1: 让请求 WWW 服务的数据包通过

Rule2: 阻挡 192.168.100.100

Rule3: 将所有的数据包丢弃

这样就无法阻挡 192.168.100.100 请求 WWW 服务了, 即第二条规则失效了

9.3.3. iptables 的表格(table)与链(chain)

1、iptables 这个防火墙软件里有多个表格(table), 每个表格都定义出自己的默认规则策略与规则, 且每个表格的用途都不相同

2、默认情况下, Linux 的 iptables 至少有 3 个表格, 包括管理本机进出的 Filter、管理后端主机(防火墙内部的其他计算机)的 NAT、管理特殊标志使用的 Mangle(较少使用)

3、表格介绍

- Filter(过滤器): 主要跟进入 Linux 本机的数据包有关, 是默认的 table
 - INPUT: 主要与想要进入 Linux 本机的数据包有关
 - OUTPUT: 主要与 Linux 本机所要送出的数据包有关
 - FORWARD: 与 Linux 本机没有关系, 它可以传递数据包到后端的计算机中, 与 NAT 的 table 的相关性较高
- NAT(地址转换): 是 Network Address Translation 的缩写, 这个表格主要用来进行来源与目的地的 IP 或 port 的转换, 与 Linux 本机无关, 主要与 Linux 主机后的局域网内计算机相关
 - PREROUTING: 在进行路由判断之前要进行规则(DNAT/REDIRECT)
 - POSTROUTING: 在进行路由判断之后要进行的规则(SANT/MASQUERADE)
 - OUTPUT: 与发送出去的数据包有关
- Mangle(破坏者): 这个表格主要是与特殊的数据包的路由标志有关

4、相关性

- <未完成>

9.3.4. 本机的 iptables 语法

1、较常用的是本机的 Filter 表格, 另一个是后端主机的 NAT 表格

9.3.4.1. 规则的查看与清除

1、<iptables>

- iptables [-t tables] [-L] [-nv]
- -t: 后面接 table, 例如 nat 或 filter, 若省略此项目, 则使用默认的 filter
- -L: 列出目前的 table 的规则
- -n: 不进行 IP 与 HOSTNAME 的反查, 显示信息的速度会快很多
- -v: 列出更多的信息, 包括通过该规则的数据包总数, 相关的网络接口
- 输出信息介绍
 - 每一个 Chain 就是每个链, Chain 所在的括号里面的是默认的策略(即没有规则匹配时采取的操作(target))
 - target: 代表进行的操作

- ACCEPT 是放行
- REJECT 是拒绝
- DROP 是丢弃
- port: 代表使用的数据包协议，主要有 TCP、UDP、ICMP3 中数据包
- opt: 额外的选项说明
- source: 代表此规则是针对哪个来源 IP 进行限制
- destination: 代表此规则是针对哪个目标 IP 进行限制

2、在 Filter 这个表格内的 INPUT、OUTPUT、FORWARD3 条规则

- INPUT 与 FORWARD 算是比较重要的管制防火链，这些链最后一条规则策略是 REJECT

3、<iptables-save>

- 由于 iptables 的上述命令的查看只是做格式化的查阅，要详细解释每个规则可能会与原规则有出入，因此，建议使用 iptables-save 这个命令来查看防火墙规则
- iptables-save [-t table]
- -t: 可以针对某些表格来输出，例如仅针对 NAT 或 Filter 等
- 输出信息介绍
 - 星号开头的指的是表格，这里为 Filter
 - 冒号开头的指的是链，3 条内建的链，后面跟策略

4、iptables [-t tables] [-FXZ]

- -F: 清除所有的已制定的规则
- -X: 清除掉所有用户"自定义"的 chain(应该说 tables)
- -Z: 将所有的 chain 的计数与流量统计都归零

5、一般来说，在重新定义防火墙的时候，都会先将规则清除掉，防火墙的规则顺序是有特殊的意义，所以应该先清除掉规则，然后一条条来设置会比较容易

9.3.4.2. 定义默认策略 (policy)

1、当数据包不在我们设置的规则之内时，该数据包的通过与否都以 Policy 的设置为准

2、一般来讲，Filter 内的 INPUT 链可以定义的比较严格一点，FORWARD 与 OUTPUT 则可以制定得轻松一点，NAT table 暂时先不理会

3、<iptables>

- iptables [-t nat] -P [INPUT,OUTPUT,FORWARD] [ACCEPT,DROP]
- -P: 定义策略(Policy)
- ACCEPT: 该数据包可接受
- DROP: 该数据包直接丢弃，不会让 Client 知道为何被丢弃
- 例子
 - iptables -P INPUT DROP
 - iptables -P OUTPUT ACCEPT
 - iptables -P FORWARD ACCEPT
 - iptables-save

9.3.4.3. 数据包的基础对比：IP、网络及接口设备

1、<iptables>

- `iptables [-A| 链名] [-i| 网络接口] [-p| 协议] \`
 - > `[-s| 来源 IP/网络] [-d| 目标 IP/网络] -j [ACCEPT|DROP|REJECT|LOG]`
- `-A| 链名:` 针对某条链进行规则的"插入"或"累加"
 - `-A:` 新增加一条规则, 该规则**增加在原规则后面**, 例如原来有 4 条规则, 使用-A 就可以加上第五条规则
 - `-I:` 插入一条规则, 如果没有指定此规则的顺序, **默认插入变成第一条规则**
 - 链: 有 INPUT、OUTPUT、FORWARD 等, 此链名称又与-i| 有关
- `-i| 网络接口:` 设置数据包进出的接口规范
 - `-i:` 数据包所进入的那个网络接口, 例如 eth0, lo 等, 需要与 INPUT 链配合
 - `-o:` 数据包所传出的网络接口, 需要与 OUTPUT 配合
- `-p| 协定:` 设置此规则适用于哪种数据包格式
 - 主要的数据包格式有: tcp、udp、icmp 以及 all
- `-s| 来源 IP/网络:` 设置此规则之数据包的来源地, 可指定单纯的 IP 或网络
 - IP: 192.168.0.100
 - 网络: 192.168.0.0/24、192.168.0.0/255.255.255.0 均可
 - **若规范为"不许"时, 加上! 即可**, 例如-s ! 192.168.100.0/24
- `-d| 目标 IP/网络:` 同-s, 只不过这里是目标的
- `-j| 后面接操作`
 - ACCEPT
 - DROP
 - REJECT
 - LOG(记录)
- 重要的原则: 没有指定的项目, 就表示该项目完全接受
 - 例如-s 和-d 不指定, 就表示来源或去向的任意 IP/网络都接受
- 例子
 - `iptables -A INPUT -i lo -j ACCEPT`
 - 上述例子没有指定-s 与-d 的规则, 表示: **不论数据包来自何处或去向哪里, 只要是 lo 这个接口, 就予以接受**
 - **这就是所谓的信任设备**
 - `iptables -A INPUT -i eth1 -j ACCEPT <==添加接口为 eth1 的网卡为信任设备`
- 想要记录某个规则的日志怎么办
 - `iptables -A INPUT -s 192.168.2.200 -j LOG <==该网段的数据包, 其相关信息就会被写入到内核日志文件中, 级/var/log/messages, 然后, 该数据包会继续进行后续的规则比对(这一点与其他规则不同)`

9.3.4.4. TCP、UDP 的规则比对: 针对端口设置

1、TCP 与 UDP 比较特殊的就是端口(port)

- 在 TCP 方面则另外有所谓的连接数据包状态, 包括最常见的 SYN 主动连接的数据包格式

2、<iptables>

- iptables [-AI 链] [-io 网络接口] [-p tcp|udp] \
 > [-s 来源 IP/网络] [--sport 端口范围] \
 > [-d 目标 IP/网络] [--dport 端口范围] \
 > --syn -j [ACCEPT|DROP|REJECT]
- --sport 端口范围：限制来源的端口号，端口号可以是连续的，例如 1024:65535
- --dport 端口范围：限制目标的端口号
- --syn：主动连接的意思
- 与之前的命令相比，就是多了--sport 以及--dport 这两个选项，因此想要使用--dport 或--sport 必须加上-p tcp 或-p udp 才行
- 例子
 - iptables -A INPUT -i eth0 -p tcp --dport 21 -j DROP <==想要进入本机 port 21 的数据包都阻挡掉
 - iptables -A INPUT -i eth0 -p tcp --sport 1:1023 \
 > --dport 1:1023 --syn -j DROP <==来自任何来源 port 1:1023 的主动连接到本机端的 1:1023 连接丢弃

9.3.4.5. iptables 外挂模块: mac 与 state

1、在 Kernel 2.2 以前使用 ipchains 管理防火墙时

- ipchains 没有数据包状态模块，因此我们必须针对数据包的进、出方向进行控制

2、iptables 可以帮我们免除这个困扰，它可以通过一个状态模块来分析这个想要进入的数据包是否为刚刚发出去的响应

3、<iptables>

- iptables -A INPUT [-m state] [--state 状态]
- -m: 一些 iptables 的外挂模块
 - state: 状态模块
 - mac: 网卡硬件地址(hardware address)
- --state: 一些数据包的状态
 - INVALID: 无效的数据包
 - ESTABLISHED: 已经连接成功的连接状态
 - NEW: 想要新建立连接的数据包状态
 - RELATED: 表示这个数据包与主机发送出去的数据包有关
- 例子：
 - iptables -A INPUT -m state \
 > --state RELATED,ESTABLISHED -j ACCEPT
 - iptables -A INPUT -m mac --mac-source aa:bb:cc:dd:ee:ff -j ACCEPT

9.3.4.6. ICMP 数据包规则的比对：针对是否响应 ping 来设计

1、<iptables>

- iptables -A INPUT [-p icmp] [--icmp-type 类型] -j ACCEPT
- --icmp-type: 后面必须要接 ICMP 的数据包类型，也可以使用代号

9.3.4.7. 超简单的客户端防火墙设计与防火墙规则存储

1、规则

- 规则归零：清除所有已经存在的规则(`iptables -F` 等)
- 默认策略：除了将 `INPUT` 这个自定义链设为 `DROP` 外，其他默认 `ACCEPT`
- 信任本机：由于 `lo` 对本机来说相当重要，因此 `lo` 必须设置为信任设备
- 回应数据包：让本机通过主动向外发出请求而响应的数据包可以进入本机 (`ESTABLISHED`、`RELATED`)
- 信任用户：这是非必要的，可在想要让本地网络的来源使用主机资源时设置

2、若要让这次修改的各种设置在下次开机时保存，就需要对 `/etc/init.d/iptables save` 这个命令加参数

9.3.5. IPv4 的内核管理功能：`/proc/sys/net/ipv4/*`

1、除了 `iptables` 这个防火墙软件之外，Linux Kernel2.6 还提供了很多内核默认的攻击阻挡机制，由于是内核的网络功能，所有相关的设置数据都是放置在 `/proc/sys/net/ipv4/` 这个目录中

2、`/proc/sys/net/ipv4/tcp_syncookies`

- 当启动 SYN Cookie 时，**主机在发送 SYN/ACK 确认数据前，会要求 Client 端在短时间内回复一个序号，这个序号包含许多原 SYN 数据包内的信息，包括 IP、port 等，若回复正确的序号，那么主机就能确定该数据包为可信的，因此会发送 SYN/ACK 数据包，否则不理会此数据包**
- 通过这一机制可以大大降低无效的 SYN 等待端口，避免 SYN Flooding 的 DoS 攻击
- `echo "1" > /proc/sys/net/ipv4/tcp_syncookies`
- 由于这个设置违反了 TCP 三次握手(**因为主机在发送 SYN/ACK 之前需要先等待 Client 的序号响应**)，所以可能造成某些服务延迟的现象

3、`/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts`

- 阻断式服务常见的就是 SYN Flooding
- 其实系统可以接受 ping 的响应，而 ping 的数据包数量可以很大，如果有许多台主机同时发送 ping 给你的主机，你的主机要么带宽被吃光，要么宕机，这种方式被称为 ping flooding
- 如何避免？取消 ICMP 类型 8 的 ICMP 数据包回应就可以
- 内核取消 ping 回应的设置值有两个
 - `/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts`: 若设置为 1，则仅有 ping broadcast 地址时才取消 ping 的回应
 - `/proc/sys/net/ipv4/icmp_echo_ignore_all`: 若设置为 1，所有 ping 都不回应

4、`/proc/sys/net/ipv4/conf/` 网络接口/*

- Linux 内核可以针对不同网络接口进行不一样的设置
- `rp_filter`: 称为逆向路径过滤(Reverse Path Filtering)，可以通过分析网络接口的路由信息，配合数据包的来源地址，来分析该数据包是否为合理
- `log_martians`: 这个设置数据可以用来启动记录不合法的 IP 来源的功能
 - 记录的数据默认放到内核放置的日志文件`/var/log/messages`
- `accept_source_route`: 建议取消

- `accept_redirects`: 建议关闭
 - 当你在同一个实体网络内架设一台路由器，但这个实体网络有两个网段，例如 `192.168.0.0/24` 和 `192.168.1.0/24`
 - 此时 `192.168.0.100` 想要向 `192.168.1.100` 传送消息时，路由器可能会传送一个 ICMP redirect 数据包告知 `192.168.0.100` 直接传送数据给 `192.168.1.100` 即可，而不需要通过路由器
 - 因为这两个网段确实在同一个实体线路上(可以直接互通)，但是确实是两个网段，因此还是无法传递消息
- `send_redirects`: 建议关闭，同上一个类似

5、**上述这些设置，可以通过`/etc/sysctl.conf`这个文件来配置**

- 配置完毕后，执行 `sysctl -p`

6、`<sysctl>`

- `sysctl` 配置与显示在`/proc/sys` 目录中的内核参数。可以用 `sysctl` 来设置或重新设置联网功能，如 IP 转发、IP 碎片去除以及源路由检查等。用户只需要编辑`/etc/sysctl.conf` 文件，即可手工或自动执行由 `sysctl` 控制的功能
- `sysctl -w variable=value`
- `sysctl -p <filename>` (default `/etc/sysctl.conf`)
- `sysctl -a`
- `-w`: 临时改变某个指定参数的值，如
`sysctl -w net.ipv4.ip_forward=1`
- `-a`: 显示所有的系统参数
- `-p`: 从指定的文件加载系统参数，如不指定即从`/etc/sysctl.conf` 中加载

9.4. 设置单机防火墙的实例

9.4.1. 规则草拟

9.5. NAT 服务器的设置

- 1、简单地说，NAT 是内部 LAN 主机的 IP 分享器
- 2、NAT 的全名是 Network Address Translation，即网络地址转换
 - NAT 可以实现 IP 分享的功能
 - NAT 还可以实现 DMZ(非军事化隔离区)的功能
 - 完全取决于 NAT 是修改来源 IP 还是目标 IP

9.5.1. 什么是 NAT?SNAT?DNAT?

- 1、若内部 LAN 有任何一台主机想要传送数据包出去(书上图 9-2)，那么这个数据包如何通过 Linux 主机传出去呢
 - **先经过 NAT table 的 PREROUTING 链**
 - 经由路由判断确定这个数据包是否要进入本机，若不进入本机，则下一步
 - 在经过 Filter table 的 FORWARD 链
 - **通过 NAT table 的 POSTROUTING 链，最后传出去**
- 2、重点在于上述 1、4 两条，重点在于修改 IP
 - POSTROUTING 修改的是来源 IP

- PREROUTING 链修改的是目标 IP
- 由于修改的 IP 不同，就称为来源 NAT(Source NAT, SNAT)或目标 NAT(Destination NAT, DNAT)

3、来源 NAT(SNAT): 修改数据包报头的来源项目

- 网络布线如书上 9-12 所示
- 客户端 192.168.1.100 这台主机要连接到 <http://tw.yahoo.com> 时，数据包报头会发生以下变化
 - 客户端所发出的数据包报头中，来源是 192.168.1.100，然后传送到 NAT 主机
 - NAT 主机的内部接口(192.168.1.2)接受这个数据包后，会主动分析报头数据，因为报头数据显示目的并非 Linux 本机，所以开始经过路由分析，将此数据包转到可以连接到 Internet 的 public IP 处
 - 由于 private IP 与 public IP 不能互通，所以 Linux 主机通过 `iptables` 的 **NATtables** 内的 **POSTROUTING** 链将数据包报头的来源伪装成 Linux 的 public IP，并且将两个不同来源(192.168.1.100 及 public IP)的数据包对应写入暂存内存当中，然后将此数据包发送出去(数据包只含有 public IP)
 - Internet 上面看到的数据包，只会知道这个数据包来自 public IP
- 当 Internet 返回数据包，会进行以下操作
 - 在 Internet 上的主机接收到这个数据包后，会将响应数据传送给 public IP 的主机
 - 当 Linux NAT 服务器收到来自 Internet 的响应数据包后，会分析该数据包的序号，并比对刚刚记录到内存当中的数据，由于发现该数据包为后端主机之前传送出去的，因此，在 NAT PREROUTING 链中，会将目标 IP 修改为后端主机，然后发现目标已经不是本机(public IP)，所以开始通过路由分析数据包流向
 - 最终传送到目标机器上去

4、所有内部 LAN 的主机都可以通过这台 NAT 服务器连接出去，而大家在 Internet 上面看到的都是同一个 IP(就是 NAT 那台主机的 public IP)，因此 NAT 最简单的功能就是类似 IP 分享器

5、目标 NAT(DNAT): 修改数据包报头的目标项目

- SNAT 主要是应付内部 LAN 连接到 Internet 的使用方式
- DNAT 则主要用在为内部主机架设可以让 Internet 访问的服务器
- 架设图见书上图 9-13
- 假设内部主机 192.168.1.210 启动了 WWW 服务器，这个服务的 port 开启在 port80，那么 Internet 上的主机如何连接到内部服务器
 - 外部主机(61.xx.xx.xx)想要连接到目标端的 WWW 服务，必须要连接到 NAT 服务器上
 - NAT 服务器已经设置好要分析出 port80 的数据包，所以当 NAT 服务器接到这个数据包后，将目标 IP 由 public IP 改为 192.168.1.210，且将该数据包相关信息记录下来，等待内部服务器响应
 - 上述的数据包在经过路由分析后，来到 private 接口处，然后通过内部的 LAN 传送到 192.168.1.210 上
 - 192.168.1.210 会响应数据给 61.xx.xx.xx，这个回应会首先传送到 192.168.1.2 上去

- 经过路由判断后，来到 NAT POSTROUTING 的链，然后通过第二步骤的记录，将来源由 192.168.1.210 改为 public IP 后，就可以传送出去了
- 上述步骤相当于是 SNAT 的反向传送，这就是 DNAT

9.5.2. 最简单的 NAT 服务器：IP 分享功能

- 1、这个 IP 分享器的功能就是 SNAT，作用就只是在 `iptables` 内的 NAT 表格中，那个路由判断后的 POSTROUTING 链所做的工作就是进行 IP 的伪装
- 2、NAT 服务器必须有一个 public IP 接口，以及一个内部 LAN 链接的 private IP 接口才行
- 3、注意：如果 public IP 取得的方式是拨号或 cable modem 时，对于配置文件 `/etc/sysconfig/network`、`ifcfg-eth0`、`ifcfg-eth1` 等，千万不要设置 GATEWAY，否则就会出现两个 default gateway，反而会出现问题
- 4、`iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -o eth0 -j MASQUERADE`
 - MASQUERADE：IP 伪装成数据包出去(-o)的那块设备上的 IP
 - 只要该数据包可以通过 eth0 传送出去，那就会自动修改 IP 的来源报头成为 eth0 的 public IP

9.5.3. iptables 的额外内核模块功能

- 1、`iptables` 提供很多好用的模块，这些模块可以辅助数据包的过滤，可以让我们节省很多 `iptables` 的规则拟定工作

9.5.4. 在防火墙后端的网络服务器上做 DNAT 设置

- 1、DNAT 用到的是 NAT table 的 Prerouting 链，不建议新手玩这个
- 2、`iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080`

9.6. firewalld

9.6.1. firewalld 简介

- 1、firewalld 是 centos7 的一大特性，最大的好处有两个：
 - firewalld 可以动态修改单条规则，而不需要像 `iptables` 那样，在修改了规则后必须得全部刷新才可以生效
 - firewalld 在使用上要比 `iptables` 人性化很多，即使不明白“五张表五条链”而且对 TCP/IP 协议也不理解也可以实现大部分功能

9.6.1.1. 关于 iptables

- 1、大部分 `iptables` 的资料都介绍说 `iptables` 包含四张表、五条链，不过实际上 `iptables` 还有第五张表——`security` 表，但是这张表需要和 `selinux` 结合使用，而 `selinux` 虽然已经发布了十多年了但是直到现在还有很多人对他的理解不够透彻，甚至有很多人会将其关闭
- 2、其实 `selinux` 的设计理念在安全上来说是非常优秀的，而且理解了其设计理念之后再去使用也没那么复杂，只不过其内置的规则是非常复杂的，等有机会学生专门给大家介绍一下 `selinux`，现在还回到 `iptables` 的五张表，他们分别是 `filter`、`nat`、`mangle`、`raw` 和 `security`
 - `filter` 表就是我们最常使用的过滤表

- nat 表主要用于数据包转发，比如局域网的电脑如果想连接互联网，那么就可以使用 nat 给转发一下
- mangle 表的规则可以对数据包进行修改，比如修改 ttl 值等
- raw 表主要是为了提高效率使用的，raw 本身的含义是指“原生的”、“未经加工的”，符合 raw 表所对应规则的数据包将会跳过一些检查，这样就可以提高效率，当然，raw 表的优先级也是最高的
- security 是跟 selinux 相关的 MAC 模式的安全过滤

9.6.1.2. firewalld 和 iptables 的关系

- 1、firewalld 自身并不具备防火墙的功能，而是和 iptables 一样需要通过内核的 netfilter 来实现
- 2、也就是说 firewalld 和 iptables 一样，他们的作用都是用于维护规则，而真正使用规则干活的是内核的 netfilter，只不过 firewalld 和 iptables 的结构以及使用方法不一样罢了

9.6.1.3. firewalld 的配置模式

- 1、firewalld 的配置文件以 xml 格式为主(主配置文件 firewalld.conf 例外)，他们有两个存储位置

- /etc/firewalld/
- /usr/lib/firewalld/

- 2、使用时的规则是这样的：当需要一个文件时 firewalld 会首先到第一个目录中去查找，如果可以找到，那么就直接使用，否则会继续到第二个目录中查找

- 3、firewalld 的这种配置文件结构的主要作用是这样的：

- 在第二个目录中存放的是 firewalld 给提供的通用配置文件，如果我们想修改配置，那么可以 copy 一份到第一个目录中，然后再进行修改
- 这么做有两个好处：
 - 首先我们日后可以非常清晰地看到都有哪些文件是我们自己创建或者修改过的
 - 其次，如果想恢复 firewalld 给提供的默认配置，只需要将自己在第一个目录中的配置文件删除即可，非常简单，而不需要像其他很多软件那样在修改之前还得先备份一下，而且时间长了还有可能忘掉之前备份的是什么版本

9.6.1.4. 配置文件结构

- 1、firewalld 的配置文件结构非常简单，主要有两个文件和三个目录

- 文件：firewalld.conf、lockdown-whitelist.xml
- 目录：zones、services、icmptypes
- 另外，如果使用到 direct，还会有一个 direct.xml 文件。我们要注意，在保存默认配置的目录 "/usr/lib/firewalld/" 中只有我们这里所说的目录，而没有 firewalld.conf、lockdown-whitelist.xml 和 direct.xml 这三个文件，也就是说这三个文件只存在于 “/etc/firewalld/” 目录中

- 2、这些文件和目录的作用

- firewalld.conf：firewalld 的主配置文件，是键值对的格式，不过非常简单，只有五个配置项

- **DefaultZone:** 默认使用的 zone, 关于 zone 学生稍后给大家详细介绍, 默认值为 public;
- **MinimalMark:** 标记的最小值, linux 内核会对每个进入的数据包都进行标记, 目的当然是为了对他们进行区分, 比如学生在前面给大家补充 iptables 五张表相关的内容时候介绍说符合 raw 表规则的数据包可以跳过一些检查, 那么是怎么跳过的呢? 这里其实就是使用的标记, 当然对数据包的标记还有很多作用。这里所设置的 MinimalMark 值就是标记的最小值, 默认值为 100, 一般情况下我们不需要对其进行修改, 但是如果我们有特殊需要的时候就可以通过对它进行修改来告诉 linux 所使用标记的最小值了, 比如我们需要给符合某条件的数据包标记为 123, 这时候为了防止混淆就需要将 MinimalMark 设置为一个大于 123 的值了;
- **CleanupOnExit:** 这个配置项非常容易理解, 它表示当退出 firewalld 后是否清除防火墙规则, 默认值为 yes;
- **Lockdown:** 这个选项跟 D-BUS 接口操作 firewalld 有关, firewalld 可以让别的程序通过 D-BUS 接口直接操作, 当 Lockdown 设置为 yes 的时候就可以通过 lockdown-whitelist.xml 文件来限制都有哪些程序可以对其进行操作, 而当设置为 no 的时候就没有限制了, 默认值为 no;
- **IPv6_rpfilter:** 其功能类似于 rp_filter, 只不过是针对 ipv6 版的, 其作用是判断所接受到的包是否是伪造的, 检查方式主要是通过路由表中的路由条目实现的, 更多详细的信息大家可以搜索 uRPF 相关的资料, 这里的默认值为 yes。
- **lockdown-whitelist.xml:** 当 Lockdown 为 yes 的时候用来限制可以通过 D-BUS 接口操作 firewalld 的程序
- **direct.xml:** 通过这个文件可以直接使用防火墙的过滤规则, 这对于熟悉 iptables 的用户来说会非常顺手, 另外也对从原来的 iptables 到 firewalld 的迁移提供了一条绿色通道
- **zones:** 保存 zone 配置文件
- **services:** 保存 service 配置文件
- **icmptypes:** 保存和 icmp 类型相关的配置文件

9.6.1.5. zone

1、firewalld 默认提供了九个 zone 配置文件: block.xml、dmz.xml、drop.xml、external.xml、home.xml、internal.xml、public.xml、trusted.xml、work.xml, 他们都保存在"/usr/lib/firewalld/zones/"目录下

2、**配置文件: /etc/firewalld/zones/[zone].xml, 每一个 zone 都有一个配置文件, 是 html 格式的文件**

3、硬件防火墙默认一般有三个区, firewalld 引入这一概念系统默认存在以下区域

4、数据包要进入到内核必须要通过这些 zone 中的一个, 而不同的 zone 里定义的规则不一样(即信任度不一样, 过滤的强度也不一样)。可以根据网卡所连接的网络的安全性来判断, 这张网卡的流量到底使用哪个 zone, 一张网卡同时只能绑定到一个 zone。大家就可以把这些 zone 想象成进入火车站的安检, 不同的入口检测的严格度不一样

5、默认的几个 zone(由 firewalld 提供的区域按照从不信任到信任的顺序排序)

- **drop**: 任何流入网络的包都被丢弃，不作出任何响应，只允许流出的网络连接。即使开放了某些服务(比如 http)，这些服务的数据也是不允许通过的
- **block**: 任何进入的网络连接都被拒绝，并返回 IPv4 的 icmp-host-prohibited 报文或者 IPv6 的 icmp6-adm-prohibited 报文。只允许由该系统初始化的网络连接
- **public(默认)**: 用以可以公开的部分。你认为网络中其他的计算机不可信并且可能伤害你的计算机，只允许选中的服务通过
- **external**: 用在路由器等启用伪装的外部网络。你认为网络中其他的计算机不可信并且可能伤害你的计算机，只允许选中的服务通过
- **dmz**: 用以允许隔离区(dmz)中的电脑有限地被外界网络访问，只允许选中的服务通过
- **work**: 用在工作网络。你信任网络中的大多数计算机不会影响你的计算机，只允许选中的服务通过
- **home**: 用在家庭网络。你信任网络中的大多数计算机不会影响你的计算机，只允许选中的服务通过
- **internal**: 用在内部网络。你信任网络中的大多数计算机不会影响你的计算机，只允许选中的服务通过
- **trusted**: 允许所有网络连接，即使没有开放任何服务，那么使用此 zone 的流量照样通过(一路绿灯)

9.6.1.6. 配置方法

1、firewalld 的配置方法主要有三种：

- **firewall-config**: 图形化工具
- **firewall-cmd**: 命令行工具
- 直接编辑 xml 文件

9.6.1.7. 名词解释

1、**target**: 目标

- 可以理解为默认行为，有四个可选值：
 - **default**
 - **ACCEPT**
 - **%%REJECT%%**
 - **DROP**
- 如果不设置默认为 **default**

2、**service**: 他表示一个服务

3、**port**: 端口，使用 port 可以不通过 service 而直接对端口进行设置

4、**interface**: 接口，可以理解为网卡接口

5、**source**: 源地址，可以是 ip 地址也可以是 ip 地址段

6、**icmp-block**: icmp 报文阻塞，可以按照 icmp 类型进行设置

7、**masquerade**: ip 地址伪装，也就是按照源网卡地址进行 NAT 转发

8、**forward-port**: 端口转发

9、**rule**: 自定义规则

9.6.1.8. 在哪个 zone 起作用(至关重要)

1、对于一个具体的请求来说应该使用哪个 zone(哪套规则)来处理呢？这个问题至关重要，如果这点不弄明白其他的都是空中楼阁，即使规则设置的再好，不知道怎样用、在哪里用也不行。

2、**对于一个接受到的请求具体使用哪个 zone, firewalld 是通过三种方法来判断的**

- source: 源地址
- interface: 接受请求的网卡
- firewalld.conf 中配置的默认 zone
- 这三个的优先级按顺序依次降低，也就是说如果按照 source 可以找到就不会再按 interface 去查找，如果前两个都找不到才会使用第三个

9.6.2. firewall-cmd

1、可选参数：

- --permanent: 表示是否存储到配置文件中(**如果存储到配置文件中这不会立即生效，需要运行 firewall-cmd --reload 来重新读取配置文件使得配置参数生效**)
- --zone: 用于指定所要设置的 zone，如果不指定则使用默认 zone
- --timeout: 不是一直生效而是生效一段时间，过期之后自动删除

2、服务器的启动/关闭/重启等

- systemctl start firewalld.service
- systemctl stop firewalld.service
- systemctl restart firewalld.service
- systemctl enable firewalld.service
- systemctl disable firewalld.service

9.6.2.1. 配置 zone

1、他是通过 firewalld.conf 配置文件的 DefaultZone 配置项来配置的，当然也可以使用 firewall-cmd 命令来配置

2、语法如下：

- firewall-cmd --set-default-zone=zone
- firewall-cmd --get-default-zone <== 获取当前默认的 zone
- **firewall-cmd --get-active-zones**
 - 查看当前起作用的 zone
 - 这个命令会返回所有绑定了 source、interface 以及默认的 zone，并会说明在什么情况下使用
- 反向查询 zone
 - firewall-cmd --get-zone-of-interface=interface
 - firewall-cmd --get-zone-of-source=source[/mask]

9.6.2.2. 配置 target

1、zone 规则中首先最重要的是 target 的设置，他默认可以取四个值： default、ACCEPT、%%REJECT%%、DROP，其含义很容易理解

2、语法如下

- `firewall-cmd --permanent [--zone=zone] --get-target`
- `firewall-cmd --permanent [--zone=zone] --set-target=target`
- 我们要特别注意，**这里的--permanent 不是可选的**，也就是说使用 `firewall-cmd` 命令也不可以让他直接生效，也需要 `reload` 才可以

9.6.2.3. 配置 source

1、只要我们将 `source` 节点放入相应的 `zone` 配置文件 (`/etc/firewalld/zones/[zone].xml`) 中就可以了，节点的 `address` 属性就是源地址

2、不过我们要注意相同的 `source` 节点只可以在一个 `zone` 中进行配置，也就是说同一个源地址只能对于一个 `zone`

3、**另外，直接编辑 xml 文件之后还需要 reload 才可以起作用**

4、语法如下

- `firewall-cmd [--permanent] [--zone=zone] --list-sources`
- `firewall-cmd [--permanent] [--zone=zone] --query-source=source[/mask]`
- `firewall-cmd [--permanent] [--zone=zone] --add-source=source[/mask]`
- `firewall-cmd [--zone=zone] --change-source=source[/mask]`
- `firewall-cmd [--permanent] [--zone=zone] --remove-source=source[/mask]`

9.6.2.4. 配置 interface

1、`interface` 有两个可以配置的位置：

- `zone` 所对应的 xml 配置文件
- 网卡配置文件(也就是 `ifcfg-*` 文件)

2、语法如下

- 配置文件修改方式与 `source` 相同，不再赘述
- `firewall-cmd [--permanent] [--zone=zone] --list-interfaces`
- `firewall-cmd [--permanent] [--zone=zone] --add-interface=interface`
- `firewall-cmd [--zone=zone] --change-interface=interface`
- `firewall-cmd [--permanent] [--zone=zone] --query-interface=interface`
- `firewall-cmd [--permanent] [--zone=zone] --remove-interface=interface`
- 另外，我们还可以在网卡配置文件中进行配置，比如可以在 `ifcfg-em1` 文件中添加下面的配置
 - `ZONE=public` <==等价于执行下面的语句
 - `firewall-cmd --zone=public --change-interface=em1`

9.6.2.5. 配置 service

1、他的配置和我们上面所介绍的 `source` 基本相同，只不过同一个 `service` 可以配置到多个不同的 `zone` 中，当然也就不需要 `--change` 命令了

2、语法如下

- `firewall-cmd [--permanent] [--zone=zone] --list-services`
- `firewall-cmd [--permanent] [--zone=zone] --add-service=service \> [--timeout=seconds]`
- `firewall-cmd [--permanent] [--zone=zone] --remove-service=service`
- `firewall-cmd [--permanent] [--zone=zone] --query-service=service`
- **--timeout:** 其含义是这样的：添加一个服务，但是不是一直生效而是生效

一段时间，过期之后自动删除

- 这个选项非常有用，比如我们想暂时开放一个端口进行一些特殊的操作(比如远程调试)，等处理完成后再关闭，不过有时候我们处理完之后就忘记关闭了，而现在的`--timeout` 选项就可以帮我们很好地解决这个问题，我们在打开的时候就可以直接设置一个时间，到时间之后他自动就可以关闭了。另外，这个参数还有更有用的用法，学生会在下面给大家讲到。当然`--timeout` 和`--permanent` 是不可以一起使用的

9.6.2.6. 配置 port

1、port 是直接对端口的操作，他和 service 非常相似

2、语法如下

- `firewall-cmd [--permanent] [--zone=zone] --list-ports`
- `firewall-cmd [--permanent] [--zone=zone] --add-port=portid[-portid]/protocol [-timeout=seconds]`
- `firewall-cmd [--permanent] [--zone=zone] \> --remove-port=portid[-portid]/protocol`
- `firewall-cmd [--permanent] [--zone=zone] \> --query-port=portid[-portid]/protocol`

9.6.2.7. 配置 icmp-block

1、icmp-block 是按照 icmp 的类型进行设置阻塞，比如我们不想接受 ping 报文就可以使用下面的命令来设置

- `firewall-cmd --add-icmp-block=echo-request`

2、语法如下

- `firewall-cmd [--permanent] [--zone=zone] --list-icmp-blocks`
- `firewall-cmd [--permanent] [--zone=zone] --add-icmp-block=icmptype \> [-timeout=seconds]`
- `firewall-cmd [--permanent] [--zone=zone] --remove-icmp-block=icmptype`
- `firewall-cmd [--permanent] [--zone=zone] --query-icmp-block=icmptype`

9.6.2.8. 配置 masquerade

1、其作用就是 ip 地址伪装，也就是 NAT 转发中的一种，具体处理方式是将接收到的请求的源地址设置为转发请求网卡的地址，这在路由器等相关设备中非常重要，比如大家很多都使用的是路由器连接的局域网，而想上互联网就得将我们的 ip 地址给修改一下，要不大家都是 192.168.1.XXX 的内网地址，那请求怎么能正确返回呢？所以在路由器中将请求实际发送到互联网的时候就会将请求的源地址设置为路由器的外网地址，这样请求就能正确地返回给路由器了，然后路由器再根据记录返回给我们发送请求的主机了，这就是 masquerade

2、语法如下

- `firewall-cmd [--permanent] [--zone=zone] --add-masquerade \> [-timeout=seconds]`
- `firewall-cmd [--permanent] [--zone=zone] --remove-masquerade`
- `firewall-cmd [--permanent] [--zone=zone] --query-masquerade`

9.6.2.9. 配置 forward-port

1、这项也非常容易理解，他是进行端口转发的，比如我们要将在 80 端口接收到 tcp 请求转发到 8080 端口可以使用下面的命令

- `firewall-cmd --add-forward-port=port=80:proto=tcp:toport=8080`
- `firewall-cmd --add-forward-port=port=80-85:proto=tcp:toport=8080`

2、语法如下

- `firewall-cmd [--permanent] [--zone=zone] --list-forward-ports`
- `firewall-cmd [--permanent] [--zone=zone] --add-forward-port=port=portid[-portid]:proto=protocol[:toport=portid[-portid]][:toaddr=address[/mask]][--timeout=seconds]`
- `firewall-cmd [--permanent] [--zone=zone] \> --remove-forward-port=port=portid[-portid]:\> proto=protocol[:toport=portid[-portid]][:toaddr=address[/mask]]`
- `firewall-cmd [--permanent] [--zone=zone] \> --query-forward-port=port=portid[-portid]:\> proto=protocol[:toport=portid[-portid]][:toaddr=address[/mask]]`

9.6.2.10. 配置 rule

1、rule 可以用来定义一条复杂的规则

2、语法如下

- `firewall-cmd [--permanent] [--zone=zone] --list-rich-rules`
- `firewall-cmd [--permanent] [--zone=zone] --add-rich-rule='rule' \> [--timeout=seconds]`
- `firewall-cmd [--permanent] [--zone=zone] --remove-rich-rule='rule'`
- `firewall-cmd [--permanent] [--zone=zone] --query-rich-rule='rule'`
- **这里的参数'rule'代表一条规则语句**

Chapter 10. 申请合法的主机名

10.1. 为何需要主机名

- 1、TCP/IP 环境只要有 IP 与正确的路由即可连接，那么**主机名的作用就是为了方便记忆**
- 2、由于 IP 是难记的东西，而且如果 IP 是类似拨号的不固定的 IP，那更伤脑筋

10.1.1. 主机名的由来

- 1、早期连上网络的计算机并不多，所以在网络上的人们就想出一个简单的办法来进行主机名与 IP 的对应，那就是在每台计算机的/etc/hosts 里面设置好主机名与 IP 的对应表，这样人们就可以通过主机名来连接到某网络上的主机了

- 随着连上 Internet 的机器越来越多，使用/etc/hosts 的方法已经搞不定了
- 因为只要一台新计算机上线，全部 Internet 上面的所有计算机都要重写 /etc/hosts，太方便了
- 这时候，DNS 就出现了

- 2、DNS(Domain Name System)

- DNS 利用类似树状目录的结构，**将主机名的管理分配在不同层级的 DNS 服务器当中**，通过分层管理的方式进行管理，所以每一台服务器记忆的信息就不会很多，而且如有变动也很容易修改
- DNS 的功能：**将计算机主机的名称转译成 IP**
- 如果想要一个主机名，那就需要通过 DNS 系统，而不是单纯地修改 /etc/hosts，而将一个主机名加入到 DNS 系统当中，重点在“授权”

10.1.2. 重点在合法授权

- 1、**错误的观点**：因为想要架设网站，所以主机需要有个主机名，因此就需要架设 DNS 服务器

- DNS 是个很庞大的架构，而且是连接在全球网络当中的
- 除非你经过注册的手续来让 DNS 系统承认你的主机名存在合法性，否则你架设的 DNS 只能说是一个练习的测试站而已，并没有实际用途

- 2、如何加入 DNS：

- 首先需要一个注册单位，并且检查出想要注册的主机名是否存在

- 3、DNS 主机名的查询方式

- 由于 DNS 查询的方式都是由上层的 ISP 提供解析授权给下层的注册者
- 因此，下层的注册者只要设置妥当，全世界的主机就会知道其设置的数据

- 4、小结

- 主机名的设计是有意义的，不可以随便设定
- 主机名要生效，需要通过注册来取得合法授权

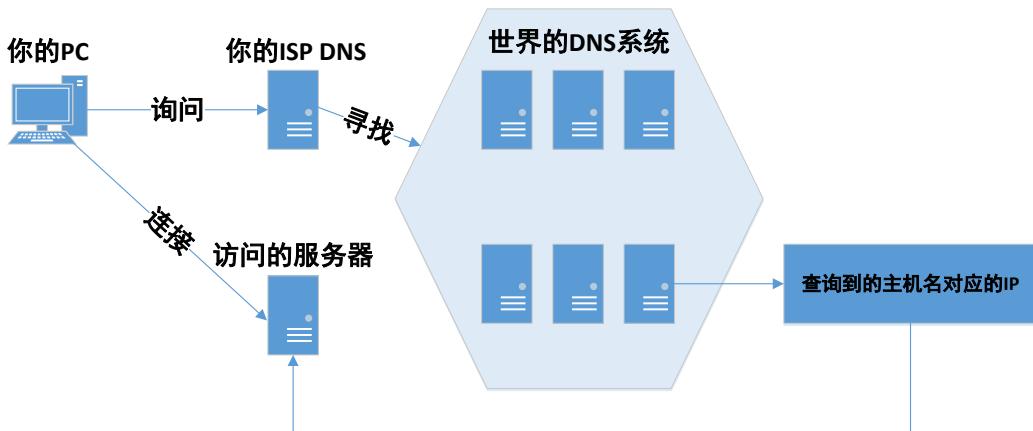


图 10-1 DNS 查询示意图

10.1.3. 申请静态还是动态 DNS 主机名

- 1、DNS 系统最大的功能就是在主机名对应 IP 的解析上
- 2、默认的 DNS 解析是用在固定 IP 于对应主机名的解析上
 - 这种情况下，在 DNS 架构下申请完主机名后，如果 IP 不改，那就永远不会有主机名相关的问题了
- 3、但是，很多小网站都以非固定 IP 来上网，更有甚者，某些 ADSL 拨号模式甚至会定时强制断线，也就是说，一段时间后，我们都得需要重新拨号上网，而每次拨号成功后取得的 IP 可见得相同
- 4、动态 DNS 服务
 - Client 就是你每次开机或者重新拨号，并取得一个新的 IP 后，会主动向 DNS Server 端提出要求，希望 Server 端变更主机名与 IP 的对应(这个步骤在每个主要的 ISP 都有提供适当的程序来给 Client 使用)
 - Server 端接受 Client 端的要求后，会先查询 Client 提供的账号密码是否正确，若正确就会立即修改 Server 本身对于你的主机名的设置
 - 因此，每次我们取得了新的 IP 后，主机名对应的 IP 也会跟着在 DNS 系统上面更新，只要别人知道你的主机名，不论你的 IP 为何，它一定可以连上你的主机

10.2. 注册一个合法的主机名

- 1、静态 DNS 主机名注册，静态 IP 对应的主机名注册网站有很多
 - www.twnic.net
 - www.netsol.com
 - www.dotster.com
 - www.godaddy.com
- 2、动态 DNS 主机名注册
 - www.no-ip.com

10.2.1. 静态 DNS 主机名注册(以 Hinet 为例)

- 1、步骤
 - 先查询所想要注册的域名是否存在

- 进入 ISP 去申请注册所想要的主机名
- 缴费，并等待主机名被启用
- 台湾地区常见的 Hinet 这个 ISP 提供的“个人域名”：.idv.tw”注册方式说明

10.2.2. 动态 DNS 主机名注册(以 no-ip 为例)

1、步骤

- 登录主网页(<http://www.no-ip.com>)，并注册一个新账号
- 进邮箱激活等等操作
- 下载一个客户端
 - wget <http://www.no-ip.com/client/linux/noip-duc-linux.tar.gz>
 - tar -zvxf noip-duc-linux.tar.gz
 - cd noip-*
 - make
 - make install
 - 选择网卡，0、1、2、3...
 - Please enter the login/email string for no-ip.com <==注册的账号或邮箱
 - Please enter the password for user '<你的输入>'<
 - Please enter an update interval <==更新的时间，分钟
 - n
 - /usr/local/bin/noip2 即可

2、<noip2>

- noip2 [-CS]
- -C：重新设置参数，亦即设置刚刚我们上面输入粗体字的内容，如果有两个以上的 no-ip 主机时，一定要用 noip2 -C 来重新设定参数文件
- -S：将目前的 noip2 的状况显示出来
- 设置开机启动
 - vim /etc/rc.d/rc.local
 - 加入以下语句： /usr/local/bin/noip2

Chapter 11. 远程连接服务器

11.1. 远程连接服务器

- 1、远程连接服务器对于管理员来说，是一个很有用的操作，它使得对服务器的管理更为方便
- 2、但与此同时，开放得让全世界都可以尝试登陆你的主机并不是个好主意，因为可能会有安全性的问题

11.1.1. 什么是远程连接服务器

1、远程连接服务器通过文字或图形接口的方式来远程登陆系统，让你在远程的中断前面登陆 Linux 主机以取得可操作主机的接口(Shell)，而登陆后的操作就像坐在系统前面一样

2、与类似 FTP 登陆不同，远程连接可以取得 Shell 进行工作

3、远程连接服务器的功能作用之一：分享 Unix-Like 主机的运算能力

4、服务器类型(Server)：有限度的开放连接

- 服务器的远程连接程序通常针对少部分系统维护者开放而已
- 除非不要，否则 Server 类型的主机不建议开放远程连接服务

5、工作站类型(Workstation)：只对内网开放

- 所谓工作站就是不提供因特网服务的主机，仅提供大量的运算能力给用户
- 必须要提供给用户登录的权限，这样大家才能使用到运算功能，此时就需要针对内部或者是特定的某些来源开放远程连接服务

11.1.2. 有哪些可供登录的类型

1、以登录的连接界面来分类，基本上有文字接口与图形接口两种

- 文字接口明文传输：Telnet、RSH 等为主，目前非常少用
- 文字接口加密：SSH 为主，已经取代上述的 Telnet、RSU 等明文传输方式
- 图形接口：XDMCP、VNC、XRD

2、在文字接口登录的连接服务器中，主要有以"明文"传送数据的 Telnet 服务器，以及利用加密技术进行数据加密再传送的 SSH 服务器，建议多使用 SSH 这种连接方式

3、数据的明文传输与加密传输

- 明文传输：数据包在网络上传输时，该数据包的内容为数据的原始格式
- 密文传输：数据包在网络上传输时，该数据包的内容为加密后的数据

11.2. 文字接口连接服务器：SSH 服务器

1、<SSH>是 Secure Shell Protocol 的简写

2、SSH 协议本身就提供两个服务器功能

- 一个就是类似 Telnet 的远程连接使用 Shell 的服务器，即俗称的 SSH
- 另一个就是类似 FTP 服务的 Stop-Server，提供更安全的 FTP 服务

11.2.1. 连接加密技术简介

1、目前常见的网络数据包加密技术通常是通过所谓的"非对称秘钥系统"来处理

的

- 主要是通过两把不一样的**公钥与私钥(Public and Private Key)**来进行加密与解密的过程，这两把钥匙的作用是提供数据加解密的，所以在同一个方向的连接中，这两把钥匙当然是需要成对的
- **公钥(Public Key):** 提供给远程主机进行数据加密的行为，也就是说，大家都能取得你的公钥来将数据加密
- **私钥(Public Key):** 远程主机使用你的公钥加密的数据，在本地端就能够使用私钥来进行解密，私钥很重要，因此私钥是不能够外流的，只能保护在自己的主机上

2、SSH 服务器端与客户端连接步骤

- **服务器建立公钥文件:**
 - 每一次启动 SSHD 服务时，该服务器会主动去找/etc/ssh/ssh_host*文件
 - 若系统刚刚安装完成，由于没有公钥文件，因此 SSH 会主动计算这些需要公钥的文件，同时也会计算出服务器自己需要的私钥文件
- **客户端主动连接要求:**
 - 若客户端想要连接到 SSH 服务器，则需要使用适当的客户端程序来连接，包括 SSH、Pietty 等客户端程序
- **服务器传送公钥文件给客户端:**
 - 接收到客户端的要求后，服务器将第一个步骤取得的公钥文件传送给客户端使用(明码传送，反正公钥就是给大家使用的)
- **客户端记录/比对服务器的公钥数据及随机计算自己的公私钥(这一点很重要，正是由于随机生成的密钥，因此每次都必须输入对方服务器主机的密码，因为服务器并没有保留连接时客户端的公钥):**
 - 若客户端第一次连接到此服务器，则会将服务器的公钥数据记录到客户端的用户主目录内的~/.ssh/known_hosts
 - 若是已经记录过该服务器的公钥数据，则客户端会去比对此次接收到的与之前的记录是否有差异
 - 若接收次公钥数据，则开始计算客户端自己的公钥数据
- **返回客户端的公钥数据到服务器端:**
 - 用户将自己的公钥传送给服务器
 - **此时服务器拥有服务器的私钥与客户端的公钥**
 - **此时客户端具有服务器的公钥与客户端的私钥**
 - 因此称为非对称秘钥系统
- **服务器接受私钥开始双向加解密:**
 - 服务器到客户端：服务器传送数据时，将用户的公钥加密后进行发送，客户端接收后，用客户端的私钥解密
 - 客户端到服务器：客户端传送数据时，将服务器的公钥加密后进行发送，服务器接收后，用服务器的私钥解密

4、建立密钥系统

- /etc/init.d/sshd restart
- systemctl restart sshd.service
- 对于 Ubuntu 系统，利用 sudo apt-get install openssh-server，来安装
 - 不知道为什么有时候在另一台虚拟机上不能连上这台 Ubuntu SSH 主机 (当时用 ps -e | grep ssh 没有发现 sshd，但是有 ssh)

- 于是卸载了 openssh-server 再重新安装 openssh-server 又好了

11.2.2. 启动 SSH 服务

- 1、在目前的 Linux Distributions 中，都是默认启动 SSH 的
- 2、直接启动就以 SSH Daemon，简称为 SSHD 来启动的

- /etc/init.d/sshd restart
- systemctl restart sshd.service

- 3、SSH 不但提供了 Shell，即 SSH Protocol 的主要目的，同时也提供了一个较为安全的 FTP Server

- 这个 SSHD 可以同时提供 Shell 与 FTP
- 而且都是架构在 port22 上面的

11.2.3. SSH 客户端连接程序——Linux 用户

- 1、<ssh>

- 直接登录远程主机的指令
- ssh [-f] [-o 参数项目] [-p 非标准端口] [账号@]IP [命令]
- -f：需要配合后面的[命令]，**不登录远程主机直接发送一个命令过去而已**
- -o：参数项目
 - ConnectTimeout=秒数：等待连接的秒数，减少等待的事件
 - StrictHostKeyChecking=[yes|no|ask]：默认是 ask，若要让 public key 主动加入 known_hosts，则可以设置为 no 即可
- -p：如果 sshd 服务启动在非标准的端口，需要使用此项目
- [命令]：不登录远程主机，直接发送命令过去，与-f 意义不太相同
- 例子
 - ssh 127.0.0.1 <==由于 SSH 后面没有加上账号，因此默认采用当前的账号来登录远程服务器
 - ssh student@127.0.0.1 <==账号为该 IP 的主机上的账号，而非本地账号哦
 - ssh student@127.0.0.1 find / &> ~/find1.log
 - ssh -f student@127.0.0.1 find / &> ~/find1.log <==会立即注销 127.0.0.1，find 在远程服务器运行

➤ **ssh -vvv [账号@]IP <==查看错误信息**

- 2、服务器公钥记录文件：~/.ssh/known_hosts

- 当你登录远程服务器时，本机会主动利用接收到的服务器的 Public Key 去比对~/.ssh/known_hosts 有无相关的公钥，然后进行下面操作
 - 若接收的公钥尚未记录，则询问用户是否记录
 - 若要记录(回答 yes)，则写入~/.ssh/known_hosts 且执行后续工作
 - 若不记录，则不写入该文件，并且退出登录工作
 - 若接收到的公钥已有记录，则比对记录是否相同
 - 若相同，则继续登录动作
 - 若不相同，则出现警告信息
 - 这是客户端的自我保护功能，避免你的服务器是别人伪装的

3、模拟 FTP 的文件传输方式：SFTP

- SSH 用在登录，SFTP 用在上传/下载

- <sftp>用法与 ssh 相似
- SFTP 使用的命令
 - 针对远程服务器主机(Server)的行为
 - cd
 - ls
 - dir
 - mkdir
 - rmdir
 - pwd
 - chgrp
 - chown
 - chmod
 - ln
 - rm
 - rename
 - exit bye quit
 - 针对本机(Client)的行为(都加上 !)
 - lcd
 - ll
 - lmkdir
 - lpwd
 - 针对资料上传/下载操作
 - put [本机目录或文件] [远程]
 - put [本机目录或文件] <==该格式, 则文件会存储到远程主机的当前目录下
 - get [远程主机目录或文件] [本机]
 - get [远程主机目录或文件] <==文件会存储在本机所在的当前目录中

4、文件异地直接复制: SCP(<scp>)

- scp [-pr] [-l 速率] file [账号@]主机:目录名 <==上传
- scp [-pr] [-l 速率] [账号@]主机:file 目录名 <==下载
- -p: 保留源文件的权限信息
- -r: 复制来源为目录时, 可以复制整个目录(含子目录)
- -l: 可以限制传输速率, 单位 Kbits/s
- scp /etc/hosts* student@127.0.0.1:~
- scp /tmp/Ubuntu.txt root@192.168.136.130:~/Desktop
- 注意, 冒号不要省略哦

11.2.4. SSH 客户端连接程序——Windows 用户

1、与 Linux 不同的是, 默认的 Windows 并没有 SSH 的客户端程序, 因此所有的程序都需要下载其他第三方软件才行, 常见的软件主要有 Pietty、PSFTP 以及 Filezilla 等

2、直接连接的 Pietty

- Windows 操作系统想要连接 SSH 服务器, 需要使用 Pietty 或 Putty 这两个

工具程序

- Pietty 除了完美兼容 Putty 之外，还提供了菜单与较为完整的文字编码
- 字符显示乱码，解决方案如下
 - 下面三个与语言编码有关的数据要相同才行
 - 文本文件本身在存档时所挑选的语言
 - Linux 程序(如 Bash 软件)本身所使用的语言(可以用 LANG 变量调整):通过 LANG 这个变量来调整
 - Pietty 所使用的语言: 通过菜单列表中的 Option 来处理

3、使用 SFTP-Server 的功能: PSFTP

- 打开程序后，输入 open IP 即可

4、图形化接口的 SFTP 客户端软件

11.2.5. SSHD 服务器详细配置

1、基本上，所有的 SSHD 服务器的详细设置都放在 /etc/ssh/sshd_config 配置文件中，/etc/ssh/sshd_config 是 SSH 的配置文件哦

- 注意，只要是设置值前面加了 #，代表该设置值的默认值，就是该值不从该配置文件读取，而是内置的一个值

2、关于 SSH Server 的整体设置

- # Port 22
 - SSH 默认使用 22 这个 Port，也可以使用多个 Port，即重复设置 Port 这个设置项目
 - 不建议修改这个
- Protocol 2
 - SSH 协议版本
 - 如果要支持旧版本，需要使用 "Protocol 2,1"
- # ListenAddress 0.0.0.0
 - 监听的主机网卡
 - 默认监听所有接口的 SSH 要求
- # PidFile /var/run/sshd.pid
 - 放置 SSHD 这个 PID 的文件
 - 上述为默认位置
- # LoginGraceTime 2m
 - 当连接上 SSH Server 之后，会出现输入密码的界面
 - 在该界面中，经过多长时间没有成功连上 SSH Server 就强迫断开连接
 - 若无单位则默认秒
- # Compression delayed
 - 指定何时开始使用压缩数据模式进行传输

3、说明主机的 Private Key 放置的文件，默认使用下面的文件即可

- # HostKey /etc/ssh/ssh_host_key <==SSH version 1 使用的私钥
- # HostKey /etc/ssh/ssh_host_rsa_key <==SSH version 2 使用的 RSA 私钥
- # HostKey /etc/ssh/ssh_host_dsa_key <==SSH version 2 使用的 DSA 私钥

4、关于登录文件的信息数据放置与 daemon 的名称

- SyslogFacility AUTHPRIV
 - 当有人使用 SSH 登录系统的时候，SSH 会记录信息，这个信息要记录在

Daemon Name 下面

- 默认是以 AUTH 来设置的，即 /var/log/secure 里面

➤ # LogLevel INFO <== 日志等级

5、安全设置项目，极重要

➤ 登录设置部分

- # PermitRootLogin yes

- 是否允许 root 登录，默认允许，建议设置成 no
- Ubuntu16 设置值是 prohibit-password，啥意思

- # StrictModes yes

- 是否让 sshd 去检查用户主目录或相关文件的权限数据
- 这是为了担心用户将某些重要文件的权限设错，可能会导致一些问题

- # PubkeyAuthentication yes

- # AuthorizedKeyFile .ssh/authorized_keys

- 是否允许用户自行使用成对的秘钥系统进行登录，仅针对 Version 2
- 至于自定义的公钥数据就放置与用户主目录下的.ssh/authorized_keys

- PasswordAuthentication yes <== 是否需要密码验证

- # PermitEmptyPasswords no <== 是否允许以空的密码登录

➤ 认证部分

- # RhostsAuthentication no

- 仅使用.rhosts 太不安全了，一定设置为 no

- # IgnoreRhosts yes

- 是否取消使用~/.ssh/.rhosts 来作为认证，一定是 yes

- # RhostsRSAAuthentication no

- 这个选项专门给 Version 1 使用的，使用 rhosts 文件在/etc/hosts.equiv
- 设置为 no

- # HostbasedAuthentication no <== 同上，专门给 Version 2 使用的

- # IgnoreUserKnownHosts no

- 是否忽略用户主目录内的~/.ssh/known_hosts 这个文件记录的主机内容

- 当然是 no

- ChallengeResponseAuthentication no

- 允许任何的密码认证，任何 login.conf 规定的认证方式均可使用

- 建议使用 PAM 模块来管理认证，因此设置为 no

- UsePAM yes

- 利用 PAM 管理用户认证

➤ 与 Kerberos 有关的设置参数

➤ 有关 X-Window 下面的相关设置

➤ 登陆后的项目

- # PrintMotd yes

- 登陆后是否显示一些信息
- 也就是打印/etc/motd 这个文件内容

- # PrintLastLog yes

- 显示上次登录的信息

- # TCPKeepAlive yes

- 当实现连接后，服务器会一直发送 TCP 数据包给客户端，用以判断对方是否一直存在连接
 - 如果连接时中间的路由器暂时停止服务几秒钟，也会让连接中断
 - 在这个情况下，任何一端死掉后，SSH 可以立刻知道，而不会有僵尸进程的出现
 - 如果网络或路由常常不稳定，那么可以设置为 no
 - UsePrivilegeSeparation yes
 - 是否使用权限较低的程序来提供用户操作
 - MaxStartups 10
 - 同时允许几个尚未登录的连接界面(尚未输入密码)
 - 已经建立连接的不计算在这 10 个当中
- 关于用户限制的设置项目(**限制不能使用 SSHD 服务器上的哪些账号或组进行登录**)
- DenyUsers * <==若是全部使用者
 - DenyUsers test <==若是部分使用者
 - DenyGroups test <==若是部分组

6、关于 SFTP 服务器与其他的设置项目

- Subsystem sftp /usr/lib/ssh/sftp-server
- # UseDNS yes
 - 一般来说，为了判断客户端来源是否正常合法，会使用 DNS 去反查客户端的主机名
 - 如果是在内网互联，设置为 no 会比较快

7、设置完毕后，重启 sshd 服务即可

- systemctl restart sshd.service

11.2.6. 制作不用密码可立即登录的 SSH 用户

1、问题引入

- 既然 SSH 可以使用 SCP 来进行网络复制，那么能不能将 SCP 的指令放置在 crontab 服务中，让系统通过 SCP 直接在后台定期进行网络复制与备份
- 默认状况下不允许此操作，因为默认状况下，必须要通过远程登陆，与 SCP 互动的输入密码才行，但是 crontab 又不会让你在终端接口输入密码，因此该程序就会一直卡住而无法在 crontab 内执行成功
- 既然 SSH 可以使用密钥系统来比对数据，并且提供用户数据加密功能，可以将 Client 产生的 Key 复制到 Server 中，因此以后 Client 登录 Server 时，由于两者在 SSH 要连接的信号传递中已经对比过 Key 了，因此可以立即进入数据传输接口中，不需要输入密码

2、实现步骤

- 客户端建立两把钥匙：
 - 私钥是解密的关键
 - 利用 ssh-keygen 来创建 Client 端的公私钥
- 客户端放置好私钥文件：
 - 将 Private Key 放在 Client 上面的用户主目录中，也就是\$Home/.ssh/
 - 将 Public Key 放在任何一个你想要用来登录的服务器端的某 User 用户主目录内的.ssh/里面的认证文件中即可完成整个程序

- 将公钥放置到服务器端的正确目录与文件中：

3、具体步骤

- **以下操作在客户端执行**
- `<ssh-keygen> [-t rsa|dsa]`
- `ssh-keygen` <==在客户端主机上以 root 身份执行此命令
- `scp ~/.ssh/id_rsa.pub [用户名]@[服务器 IP 或 hostname]:~`
- **以下操作均在服务端执行**
- 如果服务器端不存在.ssh
 - `mkdir .ssh; chmod 700 .ssh` <==权限必须是 700
- `cat id_rsa.pub >> .ssh/authorized_keys` <==在~路径下执行该命令
- `chmod 644 .ssh/authorized_keys` <==权限必须是 644
- 于是从 192.168.136.130 以 root 身份登录 192.168.136.128 的 liuye 账号就不需要输入密码了

11.2.7. 简易安全设置

- 1、其实 SSHD 并不怎么安全，很多人利用 SSH 的程序漏洞来取得远程主机 root 的权限，进而"黑掉"对方的主机
- 2、SSHD 所谓的"安全"其实指的是 SSHD 的数据是加密过的，所以它的数据在 Internet 上面传递是比较安全的，至于 SSHD 服务本身就不是那样安全了

- 如果不是必须将 SSHD 对 Internet 开放可登陆的权限的话，就尽量限制在几个小范围内的 IP 或主机名

3、安全的设置可以由以下三方面来进行

- 服务器软件本身的设置强化：/etc/ssh/sshd_config
 - 例如以下，该的方法参考 11.2.5
 - 禁止 root 这个账号使用 SSHD 服务
 - 禁止 nossh 这个组的用户使用 SSHD
 - 禁止 testssh 这个用户使用 SSHD
- TCP Wrapper 使用：/etc/hosts.allow, /etc/hosts.deny
 - 如果只想让本机以及 LAN 内的主机能够登陆，就如下操作
 - vim /etc/hosts.allow
`sshd:127.0.0.1 192.168.136.0/255.255.255.0`
 - vim /etc/hosts.deny
`sshd:ALL`
- iptables 的使用：iptables.rule, iptables.allow
 - `iptables -A INPUT -i eth0 -s 192.168.136.0/24 -p tcp --dport 22 -j ACCEPT`

11.3. 最原始图形接口：XDMCP 服务的启用

11.3.1. X Window 的 Server/Client 架构与各组件

- 1、X Window System 在运行过程中，因控制的数据不同而分为 X Server 和 X Client 两种程序

- X Server：这组程序主要负责的是屏幕画面的绘制与显示
- X Client：这组程序主要负责的是数据的运算

- 2、由于每个 X Client 都是独立存在的程序，在图形显示中会发生一些迭图的问

题，后来发展成为由一组特殊的 X Client 来管理其他 X Client 程序，这个总管就是 Window Manager

- Window Manager(WM)是一组控制所有 X Client 的管理程序，并同时提供例如任务栏、背景桌面、虚拟桌面、窗口大小、窗口移动与重迭显示等任务
- Window Manager(WM)主要由一些大型的桌面工具开发而来，常见的由 GNOME、KDE、XFCE 等

3、早期的用户在登录系统后，必须要自己先启动 X Server 程序，然后再启动个别的 Window Manager，若有其他需要，再启动额外的 X Client 即可，后来出现了所谓的 Display Manager(DM)

- Display Manager(DM)用于提供登录画面，以让用户可以通过图形接口登录
- 在用户登录后，可以通过 Display Manager 的功能区呼叫其他的 Window Manager

4、X Window System 用在网络上的方式：XDMCP

- 在网络上启动 X 的方式
 - 首先在客户端启动一个 X Server 将图形接口绘图所需要的硬件设备配置好，并且启动一个 X Server 常见的接口端口(通常是 Port 6000)
 - 然后再由服务器端的 X Client 取得绘图数据，将数据绘制而成图
 - 通过这个机制，可以在任何一台计算机中启动 X Server 来登录服务器
- 上述方法需要服务器将 X Client 程序一个个加载回来，比较麻烦
- **XDMCP(X Display Manager Control Protocol)**提供了直接通过服务器的 Display Manager 就能够提供登录的认证与加载的功能
 - XDMCP 启动后会在服务器的 UDP177 开始监听
 - 然后当客户端的 X Server 连接到服务器的 Port 177 之后，我们的 XDMCP 就会在客户端的 X Server 中放上用户输入的账号、密码的图形接口程序
 - 于是可以通过这个 XDMCP 去加载服务器提供的类似 Window Manager 的相关 X Client，即能够取得图形接口的远程连接服务器了

11.3.2. 设定 GDM 的 XDMCP 服务

1、XDMCP 协议是由 DM 程序提供的协议

- CentOS 默认的 DM 为 GNOME 所提供的 GDM
- GDM 的设置数据都放置在/etc/gdm/目录下
- 我们需要修改的配置文件其实是一个/etc/gdm/custom.conf 文件，行进如下设置
[daemon]

```
[security]
AllowRemoteRoot=yes
DisallowTCP=false
```

```
[xdmcp]
Enable=true
```

```
[greeter]
```

[chooser]

[debug]

2、netstat -tulnp

➢ udp 0 0 0.0.0.0:177 0.0.0.0:* 1671/gdm

➢ 看到类似上一行，说明 gdm 启动了

3、注意，这一小节设置的是服务器

11.3.3. 用户系统为 Linux 的登录方式

1、在不同的 X 环境下启动连接：直接用 X

- 如果客户端已经在 runlevel 5，那已经有一个 X 窗口的环境，这个环境显示的终端就称为":0"，对于 CentOS 的环境，那么该图形接口在 tty1 终端
- 如果由 runlevel 3 启动图形接口，那就是在 tty7
- 由于已经有一个 X 了，因此你必须要在另外的终端启动另一个 X 才行，那个新的 X 就称为":1"接口，其实通常就在 tty7 或 tty8
- 由于 X Server 要接收 X Client 必须要由授权才行，所以你要现在窗口接口中开放接收来自服务器的 X Client 数据

2、虽然你的客户端是以主动方式连接到服务器的 UDP Port177，但是服务器的 X Client 会主动连接到客户端的 X Server，因此，你必须开放来自服务器端主动对你的 TCP Port 6001(因为是":1"接口)的防火墙连接才行

3、<不知道为什么连不上>

11.3.4. 用户系统为 Windows 的登录方式：Xming

1、<未完成>

11.4. 华丽的图形接口：VNC 服务器

11.4.1. 默认的 VNC 服务器

1、VNC Server 会在服务器端启动一个监听用户要求的端口，一般端口号在 5901~5910 之间。当客户端启动 X Server 连接到 5901 之后，VNC Server 再将一堆预先设置好的 X Client 通过这个连接传递到客户端上，最终就能够在客户端显示服务器的图形接口了

2、默认的 VNC Server 都是独立提供给"单一"一个客户端来连接的，因此当你要使用 VNC 时，在连接到服务器去启动 VNC Server 即可

3、一般来说，VNC Server 都是手动启动，使用完毕后，再关闭即可

4、注意

- VNC Server 是服务器端需要启动的服务
- X Server 是在客户端的
- X Client 是在服务端的

5、<vncserver>

- yum install tigervnc-server
- vncserver [:号码] [-geometry 分辨率] [options]
- vncserver [-kill :号码]

- :号码: 就是讲 VNC Server 开在哪个端口, 如果是":1"则代表 VNC 5901 端口
- -geomerty: 就是分辨率, 例如 1024x768 等
- options: 其他 X 相关的选项, 例如-query localhost 之类的
- -kill: 将已经启动的 VNC 端口删除
- 例子:
 - vncserver :3 <==将 VNC Server 启动在 5903 端口
 - 第一次开启时, 期间需要输入两次密码, 该密码为客户端连接时需要的密码
 - 密码至少需要 6 个字符
 - 根据使用 VNC Server 的身份, 将刚刚建立的密码放置于该账号用户主目录下: ~/.vnc/passwd
 - 若该文件已经存在, 则不会出现建立密码的过程
 - 当客户端连接成功后, 服务器将会传送~/.vnc/startx 内的 X Client 给客户端
 - vncpasswd <==修改密码
 - iptables -A INPUT -i eth0 -s 192.168.136.0/24 -p tcp \
 > --dport 5900:5910 -j ACCEPT <==放行这些端口的防火墙规则
 - **vncserver -kill :3** <==使用完毕后记得关闭该服务哦
 - **vncserver -list** <==查看服务开放情况

11.4.2. VCN 的客户端连接软件

1、Linux 客户端程序: VNC Viewer

- yum install tigervnc
- vncviewer **192.168.136.130:3** <==进行连接
 - **主机 IP 与端口号之间只有一个冒号, 没有任何空格**
 - **No useful address for host** <==第一次连接出现了这样的错误
 - 经过检查发现是防火墙的原因, 每台主机最好详细制定 iptables 的防火墙规则

2、Windows 客户端程序: RealVNC

1、www.realvnc.com 下载 Windows 版本即可

- 会同时安装 VNC Server 与 VNC Viewer
- 启动 VCN Viewer
- 以"IP:端口号"作为 VNC Server 的 IP
 - 192.168.200.101:1
 - **注意 IP 与端口号之间只有一个冒号, 没有任何空格**

11.4.3. VNC 搭配本机的 XDMCP 画面

1、若需要使用 VNC 来搭配 XDMCP 的输入时

- 必须已经启动 XDMCP

2、感觉没啥用

11.4.4. 开机就启动 VNC Server 的方法

1、不要将 VNC Server 的指令写入到/etc/rc.d/rc.local 文件中，否则可能会产生 localhost 无法登陆的问题

2、步骤

- vim /etc/sysconfig/vncservers
 - VNCSEVER=“1:liuye”
 - VNCSEVERARGS[1]=“-query localhost”
 - 上述两行的 1 指的是 5901 端口
- systemctl restart vncserver.service???(好像没有这个服务)
- systemctl enable vncserver.service???(好像没有这个服务)

3、<未完成>

11.4.5. 同步的 VNC：可以通过图示同步教学

- 1、由于 Linux 本身提供多个 VNC Server，它们是各自独立的
- 2、<未完成>

11.5. 仿真的远程桌面系统：XRDP 服务器

- 1、XDMCP 与 VNC 数据原则上都没有加密，因此仅适合局域网内操作
- 2、<未完成>：yum 上没有改 XRDP 软件，需要自己配置--

11.6. SSH 服务器的高级应用

11.6.1. 在非标准端口启动 SSH(非 Port22)

- 1、很多 ISP 在入口处就已经将 Port22 关闭了
 - 很多 Cracker 会使用扫描程序乱扫整个 Internet 的端口漏洞，Port22 就是一个经常被扫描的端口
 - 为了杜绝这个问题，ISP 会先帮你把关，将 Port22 关闭
- 2、我们可以将 SSH 放在非标准的端口，如此一来，Cracker 不会扫描到该端口，你的 ISP 又没有对该端口进行限制，那就能够正常使用 SSH 了
- 3、步骤
 - 设定 SSH 在 Port22 以及 23 两个端口监听
 - 修改/etc/ssh/sshd_config
 - **systemctl restart sshd.service** <== 启动 23 端口失败了，查找 setroubleshoot 的相关信息"cat /var/log/messages | grep setroubleshoot"
 - 由于 CentOS 将 SSH 规范 Port 仅能启动于 22，所以会出现一个 SELinux 错误
 - 按照 setroubleshoot 提示的修改方式一步步修改即可
- 4、非标准端口的连接方式
 - ssh -p 23 root@localhost

11.6.2. 以 rsync 进行同步镜像备份

- 1、常用的备份指令包括：tar、dd、cp 等
- 2、rsync 可以作为一个相当棒的异地备份系统的备份指令，rsync 可以达到类似镜像的功能
 - rsync 最早是想要取代 rcp 这个指令
 - rsync 不但传输的速度快，而且在传输时，可以比对本地端与远程主机欲复制的文件内容，而仅复制两端有差异的文件而已
- 3、rsync 至少可以通过三种方式来工作
 - 在本机上直接运行，用法几乎与 cp 一模一样
 - rsync -av /etc /tmp
 - 通过 rsh 或 ssh 的信道在 Server/Client 之间进行传输
 - rsync -av -e ssh user@rsh.server:/etc /tmp
 - 直接通过 rsync 提供的服务(daemon)来传输，此时，rsync 主机需要启动 873 port

- 必须要在 Server 端启动 rsync
- 必须编辑/etc/rsyncd.conf 配置文件
- 必须设置定义 Client 端连接的密码数据
- 在 Client 端可以利用: rsync -av user@hostname:/dir/path /local/path

4、<rsync>

- rsync [-avrlptgoD] [-e ssh] [user@host:/dir] [/local/path]
- -v: 查看模式, 可以列出更多信息, 包括镜像文件名等
- -q: 与-v相反, 安静模式, 略过正常信息, 仅显示错误信息
- -r: 递归复制, 可以针对目录处理
- -u: 仅更新, 若目标文件较新, 则保留新文件不会覆盖
- -l: 复制链接文件的属性, 而非链接的目标文件内容
- -p: 复制时, 连同属性(permission)一起复制
- -g: 保存源文件的属组
- -o: 保存源文件的属主
- -D: 保存源文件的设备属性(device)
- -t: 保存源文件的时间参数
- -l: 忽略更新事件(mtime)的属性, 文件对比上会比较快速
- -z: 在数据传输时, 加上压缩的参数
- -e: 使用的协议, 例如 ssh 通道为-e ssh
- **-a: 相当于-rlptgoD, 最常用**

11.6.3. 通过 SSH 通道加密原本无加密的服务

1、SSH 这个通道可以加密, 那么其他服务能不能通过这个 SSH 进行数据加密来传递信息呢

2、假设服务器在 Port 5901 启动了 VNC 服务, 客户端则使用 VNC Viewer 连接到服务器上的 Port 5901

- 现在在客户端计算机上启动一个 5901 端口
- 然后通过本地端的 SSH 连接到服务器的 SSHD 中
- 服务器的 SSHD 再去连接服务器的 VNC Port 5901
- 整个流程如下图

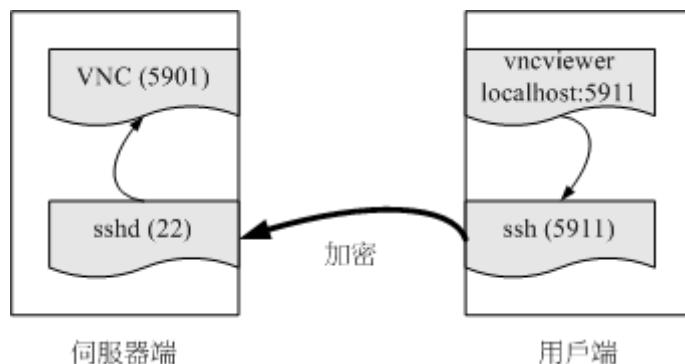


图 11-1 通过本地端的 SSH 加密连接到远程服务器的配置模型

3、具体步骤<ssh>

- ssh -L [本地端口]:127.0.0.1:[远程端口] -N 远程主机

- `ssh -L 5911:127.0.0.1:5901 -N 192.168.136.130` <==建立了从客户端 5911 端口到远程 5901 端口的通道，建立时要输入密码(如果已经制作了非密码的登录文件，那就不需要)
- `vncviewer localhost:5911` <==直接尝试连接客户端的 5911 端口，由于上一条已经开辟了通道，因此会直接连通到服务端的 5901 端口

11. 6. 4. 以 SSH 通道配合 X Server 传递图形界面

Chapter 12. DHCP 服务器

12.1. DHCP 的工作原理

1、DHCP(Dynamic Host Configuration Protocol)

12.1.1. DHCP 服务器的用途

1、回顾

- 要配置好一个网络环境，使计算机可以连上 Internet，那计算机一定要有 IP、netmask、broadcast、gateway、DNS IP 等网络参数才行
- 其中 IP、netmask、broadcast、gateway 都可以在/etc/sysconfig/network-scripts/ifcfg-eth[0-n]这个文件里面定义
- DNS 服务器的地址在/etc/resolv.conf 文件中定义
- 只要上述几个项目配置正确，那么计算机连接 Internet 就没问题了

2、如果一个局域网内有很多主机，如何进行配置

- 直接手动配置好每一台计算机
- 将所有主机用户集中起来，教他们如何配置
- 利用一台主机来自动分配所有的网络参数给局域网内的计算机

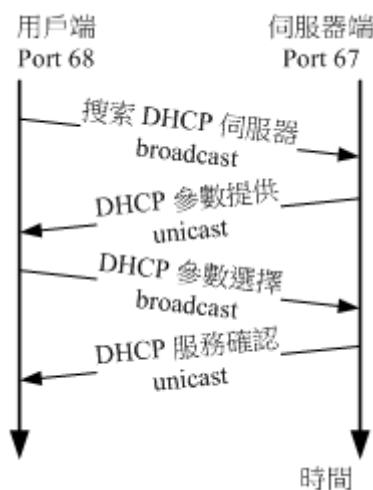
3、DHCP 服务器最主要的工作，就是实现上面第三种方案

- 自动将网络参数正确地分配给网络中的每台计算机，让客户端计算机可以在开机的时候就立即自动配置好网络的参数值
- 这些参数值包括 IP、netmask、network、gateway 与 DNS 地址

12.1.2. DHCP 协议的工作方式

1、DHCP 通常是在用于**局域网内**的一个通信协议

- 它主要是通过客户端发送广播数据包给整个物理网段内的所有主机，若局域网有 DHCP 服务器时，才会响应客户端的 IP 参数要求
- 所以 DHCP 服务器与客户端应该在同一个物理网段内



图表 12-1 DHCP 数据包在服务器与客户端之间的交互情况示意

3、客户端取得 IP 参数的程序可以简化如下

- **客户端：利用广播数据包发送搜索 DHCP 服务器的数据包**
 - 若客户端网络设置使用 DHCP 协议取得 IP，则当客户端开机或者是重新

启动网卡时，客户端主机会发送出查找 DHCP 服务器的 UDP 数据包给所有物理网段内的计算机

- 此时数据包的目标 IP 会是 255.255.255.255
 - 因此一般主机接收到这个数据包后直接丢弃
 - 若局域网内有 DHCP 服务器时，则开始后续行为
- **服务器端：提供客户端网络相关的租约以供选择**
 - DHCP 服务器在接受到客户端的要求后，会针对这个客户端的硬件地址 (MAC)与本身的设置数据来进行以下工作
 - 到服务器的日志文件中查找该用户之前是否曾经租用过某个 IP，若有且该 IP 目前无人使用，则提供此 IP 给客户端
 - 若配置文件针对该 MAC 地址提供特定的固定 IP(Static IP)时，则提供该固定 IP 给客户端
 - 若不符合上述两个条件，则随机选取当前没有被使用的 IP 参数给客户端，并记录下来
 - 总之，服务器端会针对客户端的要求提供一组网络参数租约给客户端选择，此时，客户端尚未拥有 IP，因此在服务器端响应的数据包中，主要是针对客户端的 MAC 地址来给予回应的
- **客户端：决定选择 DHCP 服务器提供的网络参数租约并向服务器确认**
 - 由于局域网内可能并非仅有一台 DHCP 服务器，但客户端仅能接受一组网络参数租约
 - 因此客户端需要选择是否要认可该服务器提供的相关网络参数的租约
 - 当决定好使用此服务器的网络参数租约后，客户端便开始使用这组网络参数来配置自己的网络环境
 - 此外，客户端也会发送一个广播数据包给所有物理网段内的主机，告知已经接受该服务器的租约，若此时有两台以上的 DHCP 服务器，则这些没有被接受的服务器会收回该 IP 租约
 - 被接受的 DHCP 服务器会继续进行下面的动作
- **服务器端：记录该次租约行为并向客户端发送相应数据包信息以确认客户端的使用**
 - 当服务器收到客户端的确认后，服务器会回送确认的响应数据包，并告知客户端这个网络参数租约的期限，并且开始租约计时
 - 以下几种情况，该次租约会到期而被解约
 - **客户端脱机**：关闭网络接口(ifdown)、重新启动(reboot)、关机(shutdown)等行为，都算是脱机状态，这个时候，Server 端就会收回 IP
 - **客户端租约到期**：DHCP Server 端发放的 IP 有使用的期限，客户端使用这个 IP 到达规定的事件，而且没有重新提出 DHCP 的申请时，Server 端就会将该 IP 收回，这个时候就会造成断线。客户端可以向 DHCP 服务器再次要求分配 IP

4、DHCP 服务器给予客户端固定或动态的 IP 参数

- 服务器会比较客户端的 MAC 硬件地址，并判断该 MAC 是否需要给予一个固定 IP
- 固定(Static IP)
 - 只要计算机的网卡不换掉，MAC 就肯定不会变
 - DHCP 可以根据 MAC 来给予固定 IP 参数租约

- 这种情况比较适合**当这台客户端计算机需要用来作为网络内的一些服务器主机**的情况
- 如何查看 MAC 地址
 - ifconfig
 - <arp> -n
- 动态(Dynamic) IP
- 除非局域网内的计算机有可能用来作为主机之用，必须设置为固定 IP，否则使用动态 IP 的设置比较简单

5、注意

- IP 只有 Public IP 与 Private IP 两种
- 静态 IP、动态 IP、虚拟 IP 等都是从获取 IP 的方式这个角度来分类的

6、关于租约所造成的问题与租约期限

- 期限的设定最大的好处就是避免 IP 被某些客户端一直占用着，但客户端确实 Idle(闲置)状态
- 比方有 150 个 IP，共 200 个人同时要使用，有了租约期限，那么最后 50 个人才有机会使用 IP
- DHCP 客户端程序大多会主动依据租约时间去重新申请 IP

7、多台 DHCP 服务器在同一物理网段的情况

- 先抢先得，DHCP 也是如此

12.1.3. 何时需要架设 DHCP 服务器

1、关于架设 DHCP 的原则性问题

- 使用 DHCP 的时机
 - 具有相当多移动设备的场合
 - 局域网内计算机的数量相当多
- 不建议使用 DHCP 主机的时机
 - 在网络中的计算机，有很多机器其实是作为主机的用途，很少有用户需求，那就没有必要架设 DHCP
 - 只有 3-4 计算机时，架设 DHCP 只能拿来当练习，并没有很大效益
 - 当网卡不支持 DHCP 协议时
 - 用户计算机水平都很高时，也没有必要架设 DHCP

12.2. DHCP 服务器端的配置

- 1、目前市面上的 IP 路由器非常便宜，而且 IP 路由器本身就含有 DHCP 的功能
- 2、下面介绍搭建简单的 DHCP 服务器

12.2.1. 所需软件与文件结构

- 1、所需的软件名称就是 dhcp，用 yum 进行安装，然后 rpm -ql dhcp 来查看这个软件提供了哪些文件
- 2、重要的配置文件如下

- /etc/dhcp/dhcpd.conf: DHCP 服务器的主要配置文件
- /usr/sbin/dhcpd: 启动 dhcp daemon 的脚本文件
- /var/lib/dhcp/dhcpd.leases: DHCP 服务器端与客户端租约建立的起始与到

期日就记录在这个文件之中

12.2.2. 主要配置文件/etc/dhcp/dhcpd.conf 的语法

1、该文件必须注意以下规范

- #为注释符号
- 除了右括号)后面之外，其他的每一行配置最后都要以";"结尾
- 配置项目的语法形式主要是<参数代号><配置内容>，例如
 - default-lease-time 259200;
- 某些项目必须以 option 来定义，基本形式为"option<参数代码><配置内容>"，例如
 - option domain-name "your.domain.name"

2、/etc/dhcp/dhcpd.conf 该文件中的配置主要分为两大项目

- 服务器运行的全局设置(Global)
- IP 分配设置(动态或固定)

3、全局设置(Global)

- 假设 **dhcpd** 管理的是只有一个子网的局域网，除了 IP 之外的许多网络参数就可以放在全局设置的区域中，包括租约期限、DNS 主机的 IP 地址、路由器的 IP 地址以及动态 DNS 更新的类型等
- 当固定 IP 以及动态 IP 没有定义到某些值时，则以全局设置为准
- **default-lease-time <时间>**: 默认的租约时间，用户没有设定特别要求租约时间的话，就以此为默认的租约事件，单位秒
- **max-lease-time <时间>**: 最大租约事件，当用户要求的租约时间超过次值时，以此值为准
- **option domain-name <域名>**: 当你要查找主机时名时，DNS 系统会主动帮你把在所要查找的主机名后加上这个域名后缀
- **option domain-name-servers IP1、IP2**
 - 这个参数值可以修改客户端的/etc/resolv.conf 文件，即 nameserver 后面接的 DNS IP
- **ddns-update-style** 类型
 - 因为 DHCP 客户端所取得 IP 通常是一直变动的，所以某台主机的主机名与 IP 的对应就很难处理
 - 此时 DHCP 可以通过 ddns 来更新主机名与 IP 的对应关系
- **ignore client-updates**: 与上一个设置值有关
- **option routers <路由器的地址>**: 设定路由器的 IP 地址

4、IP 分配设置(动态或固定)

➤ **由于 dhcpd 主要是针对局域网来分配 IP 参数的，因此在设置 IP 之前，我们需要指定一个局域网才行**

- **subnet <NETWORK_IP> netmask <NETMASK_IP> {...}**
- **range IP1 IP2**
 - 在这个局域网中，给予一个连续的 IP 地址段用来发放给客户端使用
 - IP1 IP2 指范围，闭区间
 - range 192.168.100.101 192.168.100.200
- **host 主机名 {...}**
 - 这个 host 就是指定固定 IP 对应到固定 MAC 的设置值

- 主机名可以自己给予
- 大括号内就需要指定 MAC 与固定 IP 了
 - hardware ethernet 硬件地址
 - fixed-address IP 地址

12.2.3. 一个局域网内的 DHCP 服务器的设置案例

1、/etc/dhcp/dhcpd.conf 的详细内容

```

ddns-update-style none;
ignore client-updates;
default-lease-time 259200;
max-lease-time 518400;
option routers 192.168.200.254; //参考 8.2.3 的配置，及 192.168.200.0/24
子网的路由地址
option domain-name "centos.liuye";
option domain-name-servers 114.114.114.114;

subnet 192.168.200.0 netmask 255.255.255.0 {
    range 192.168.200.101 192.168.200.200;

    host Linux2{
        hardware ethernet 00:0c:29:04:28:21;
        fixed-address 192.168.200.66;
    }
}

```

2、由于现在设置的是 192.168.200.0/24 这个子网，但是万一这台路由器还有一个网卡连接着另一个网段，例如 192.168.100.0/24，那么这个网段的主机发送出 DHCP 数据包的请求，取得的 IP 仍然是 192.168.200.0/24 这个子网的 IP

- 可以通过设置/etc/sysconfig/dhcpd 配置文件，并写入以下语句
- DHCPDARGS=<网卡接口> <==表明只响应指定网卡接口的请求
- 但是在 CentOS 5.x 以后已经不需要这样配置了，因为 DHCP 会主动分析服务器的网段与实际的 dhcpd.conf 配置，若无法吻合，就会有错误提示

12.2.4. DHCP 服务器的启动与查看

1、启动

- systemctl restart dhcpcd.service
- 若配置文件语法错误的话，会有相关提示，按照提示进行更改便是
- dhcpcd 使用的端口是 port 67，并且启动的结果会记录在/var/log/messages 文件内
- tail -n 30 /var/log/message，若发现以下内容则说明成功
 - Wrote 0 deleted host decls to leases file.
 - Wrote 0 new dynamic host decls to leases file.
 - Wrote 0 leases to leases file.

12.2.5. 内部主机的 IP 对应

- 1、就是把所有的静态 IP 和动态 IP 与 hostmae 的对应关系都写入/etc/hosts
- 2、有 DNS 服务器的话就更好了

12.3. DHCP 客户端的设置

12.3.1. 客户端是 Linux

1、/etc/sysconfig/network-scripts/ifcfg-eno16777736

- DEVICE="eno16777736"
- HWADDR="00:0c:29:04:28:21"
- NM_CONTROLLED="no"
- ONBOOT="yes"
- **BOOTPROTO=dhcp**

2、将整个网络重启即可，不要用 ifdown 和 ifup，因为还有默认路由要设置

- 如果执行的结果找到了正确的 DHCP 主机，那么以下文件会被修改
 - /etc/resolv.conf
 - 路由表
 - /var/lib/dhclient/dhclient--eno16777736.lease <==这个文件记录了该网卡曾经要求过的 DHCP 信息，不同的网卡会有不同的文件(后缀是网卡接口的名称)

12.3.2. 客户端是 Windows

- 1、<未完成>：

12.4. DHCP 服务器端的高级查看与使用

1、如果需要管理的是几十部甚至是上百部计算机时，总是希望能够根据座位来进行 IP 的分配，因此固定 IP 配合 MAC 就显得很重要了

12.4.1. 检查租约文件

1、客户端会主动记录租约信息，服务端也会记录

- /var/lib/dhcpd/dhcpd.leases

12.4.2. 让大量 PC 都具有固定 IP 的脚本

- 1、<未完成>

12.4.3. 使用 ether-wake 实现远程自动开机 (remote boot)

1、既然已经知道了客户端的 MAC 地址了，如果客户端主机支持一些电源标准，并且该客户端主机所使用的网卡与主板支持网络唤醒功能，我们就可以通过网络来让客户端计算机开机

2、首先要进行如下处理

- 需要在 BIOS 里面设置"网络唤醒功能"
- 必须要让这台主机接上网线，并且电源是接通的
- 将这台主机的 MAC 地址抄下来，关机等待网络唤醒

3、然后在 DHCP 服务器上面安装 net-tools 这个软件，就会得到 ether-wake 这个命令

- ether-wake -i eth1 11:22:33:44:55:66
- ether-wake -u <==查阅更多功能

12.4.3.1. DHCP 与 DNS 的关系

1、如果局域网内有很多 Linux 服务器时，需要将 Private IP 加入到每台主机的 /etc/hosts 里面，这样连接阶段的等待事件才不会有超时或者是等待太久的问题

2、此时在局域网内搭建一台 DNS 服务器负责主机名的解析就很重要

- 于是，有了 DNS 服务器的帮忙进行主机名的解析，就不需要修改/etc/hosts 了
- 未来的新机器或者新加入的计算机也不需要改写任何网络参数，这样维护会轻松很多
- 我们应该在 DHCP 服务器主机上面再安装一个 DNS 服务器，提供内部计算机的名称解析

3、DHCP 响应速度与有网管功能的 Switch 的设置问题

- <未完成>

Chapter 13. 文件服务器之一：DFS 服务器

13.1. NFS 的由来与功能

1、NFS 这个通过网络共享文件系统的服务在搭建的时候很简单，其最大的特点在于"权限"

- 在客户端与服务器端必须具备相同的账号才能够访问某些目录或文件
- NFS 的启动需要通过所谓的远程过程调用(RPC)，也就是说，我们并不是只要启动 NFS 就可以了，还需要启动 RPC 这个服务才行

13.1.1. 什么是 NFS (Network File System)

1、NFS 最大的功能就是可以通过网络，让不同的机器，不同操作系统可以共享彼此的文件(**share files**)

- NFS 服务器可以让 PC 将网络中的 NFS 服务器共享的目录挂载到本地端的文件系统中
- 在本地端的系统中看来，那个远程主机的目录就好像是自己的一个磁盘分区一样，使用上相当便利

2、当 NFS 服务器配置好共享出来的/home/sharefile 这个目录后，其他的 NFS 客户端就可以将这个目录挂在到自己的文件系统的某个挂载点

3、基本上 NFS 这个服务的端口开在 2049

- 但是由于文件系统非常复杂，因此 NFS 还有其他的程序去启动额外的端口，这些额外的端口是非确定的，因为默认 NFS 用来传输的端口是随机选择的，小于 1024 的端口
- 那么客户端怎么知道使用哪个端口呢？此时就需要远程过程调用(Remote Procedure Call, RPC)协议来辅助来了

13.1.2. 什么是 RPC (Remote Procedure Call)

1、由于 NFS 支持的功能非常多，而不同的功能都会使用不同的程序来启动，每启动一个功能就会启用一些端口来传输数据

- 因此，NFS 的功能对应的端口并不固定，而是随机取用一些未被使用的小于 1024 的端口用于传输
- 但如此一来，又会产生客户端连接服务器的问题，因为客户端需要知道服务器端的相关端口才能够连接

2、RPC 最主要的功能就是：**指定每个 NFS 功能对应的 port number，并且通知给客户端，让客户端可以连接到正确的端口上去**

- 当服务器启动 NFS 时会随机选取数个端口，并主动向 RPC 注册，因此 RPC 可以知道每个端口对应的 NFS 功能
- 然后 RPC 又以固定使用 111 来监听客户端的需求并向客户端响应正确的端口

3、注意：

- 启动 NFS 之前，RPC 就要先启动，否则 NFS 会无法向 RPC 注册
- RPC 若重启，原来注册的数据会不见，因此 RPC 重启后，它所管理的服务都需要重新启动以向 RPC 注册

4、NFS 服务器响应客户端需求的过程

- 客户端会向服务器的 RPC(port 111)发出 NFS 文件访问功能的查询要求
- 服务器端找到对应的已注册的 NFS daemon 端口后，会通知给客户端
- 客户端了解正确的端口后，就可以直接与 NFS daemon 连接

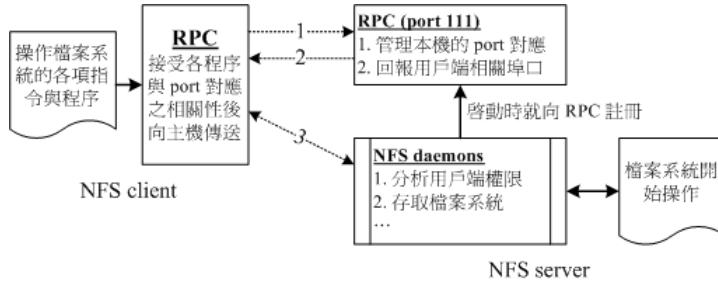


图 13-1 NFS 服务及文件系统操作的相关性

5、从上图可知，不论是客户端还是服务器端，要使用 NFS 时，两者都需要启动 RPC 才行

6、NFS 必须在 RPC 存在时才能成功地提供服务，因此我们称为 NFS 为 RPC Server 的一种，类似的还有 NIS(Network Information Service)

13.1.3. NFS 启动的 RPC daemons

1、NFS 服务器在启动的时候需要向 RPC 注册，所以 NFS 服务器也称为 RPC Server 之一

- NFS 服务器主要的任务是进行文件系统的共享，而文件系统的共享是与权限有关的
- NFS 服务器启动时至少需要两个 daemon
 - 一个管理客户端是否能够登陆的问题
 - 一个管理客户端能够取得的权限
 - 如果要管理 quota 的话，那 NFS 还需在加载其他 RPC 程序

2、以功能较单纯的 NFS 服务器来说，需要启动以下 daemon

- rpc.nfsd: 最主要的 NFS 服务提供程序
 - 其主要功能就是管理客户端是否能够使用服务器文件系统挂载信息等，其中还包含判断这个登录用户 ID
- rpc.mountd: 其最主要的功能就是在管理 NFS 的文件系统
 - 当客户端顺利通过 rpc.nfsd 登录服务器后，在它可以使用 NFS 服务器提供的文件之前，还会经过文件权限(就是-rwxrwxrwx 与 owner, group 那几个权限)的认证程序
 - 它会去读取 NFS 的配置文件/etc(exports 来比对客户端的权限，当通过这一关之后客户端就可以取得使用 NFS 文件的权限了
 - 这个 daemon 也是我们用来管理 NFS 共享目录的权限与安全设置的地方
- rpc.lockd(非必要): 这个 daemon 用于管理文件的锁定(lock)方面，同步的问题
 - 既然共享的 NFS 文件可以让客户端使用，那么当多个客户端同时尝试写入某个文件时，就可能对该文件造成一些问题
 - rpc.lockd 可以用来客服这些问题，但必须要同时在客户端与服务器端都开启才行
 - rpc.lockd 常与 rpc.statd 同时启动
- rpc.statd(非必要): 这个 daemon 可以用来检查文件的一致性

- 与 rpc.lockd 有关
- 若发生因为客户端同时使用同一文件造成文件可能有所损毁时，
rpc.statd 可以用来检测并尝试恢复该文件
- 与 rpc.lockd 一样，这个功能必须要在服务器端与客户端都启动才会生效

13.1.4. NFS 的文件访问权限

1、问题引入：

- 假设我们在 NFS Client 1 上面以 dmtsai 这个用户身份去访问 /home/data/sharefile/ 这个来自 NFS Server 所提供的文件系统
- 那么 NFS Server 提供的文件系统会让我们以什么身份去访问？是 dmtsai 还是其他？

2、NFS 本身的服务并没有进行用户身份验证

- 所以当客户端以 dmtsai 的身份想要访问服务器端的文件系统时，服务器端会以客户端的用户 UID 与 GID 身份来尝试读取服务器端的文件系统
- 于是产生了一个问题：如果客户端与服务器端的用户身份并不一致怎么办
- 文件系统的 inode 所记录的属性为 UID、GID，而非账号名与属组名，一般 Linux 主机会主动以自己的 /etc/passwd、/etc/group 来查询对应的用户名、组名
- 当 dmtsai 进入到该目录后，会参照 NFS Client1 的用户名与组名，但是由于该目录的文件主要来自 NFS Server，因此会出现以下几种情况
 - **NFS Server/NFS Client 刚好有相同的账号与用户组**
 - 此时直接以 dmtsai 的身份访问服务器所提供的共享文件系统
 - **NFS Server 的 501 这个 UID 对应的账号为 liuye**
 - 若 NFS Server 上的 /etc/passwd 里面的 UID 501 的用户名是 liuye，则客户端 dmtsai 可以访问服务器端的 liuye 这个用户的文件，**因为两者具有相同的 UID 而已**
 - **NFS Server 并没有 501 这个 UID**
 - 在服务器端并没有 501 这个 UID 存在，则此时 dmtsai 的身份在该目录下会被压缩成匿名用户，一般 NFS 的匿名者会把 65534 作为其 ID
 - 但也会有特殊情况，例如在服务器端共享 /tmp 的情况下，dmtsai 的身份还是保持 501，但建立的各项数据在服务器端看来，就会属于无属主的数据
 - **如果用户身份是 root**
 - 每个 Linux 主机都有 UID 为 0 的 root
 - 在默认情况下，root 的身份会被主动压缩成匿名用户

3、总之，客户端用户能做的事情是与 UID 以及 GID 有关的，当客户端与服务器端的 GID 及账号的对应不一致时，可能就会造成文件系统使用上的混乱，这是 NFS 文件系统在使用上的一个很重要的弊端

4、在实际客户端以 NFS 使用服务器端的文件系统，还需要具备

- NFS 服务器已经开放可写入的权限(与 /etc/exports 设置有关)
- 实际的文件权限具有可写入(w)的权限

5、当满足了以下条件，才有文件的可写入权限

- 用户账号，即 UID 的相关身份
- NFS 服务器允许写入的权限

- 文件系统确实具有 w 的权限

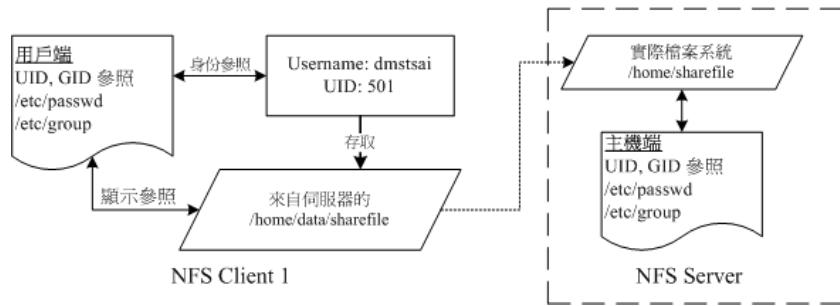


图 13-2 NFS 的服务器端与客户端的用户身份确认机制

13.2. NFS Server 端的配置

13.2.1. 所需要的软件

1、RPC 主程序: rpcbind

- NFS 可以被视为一个 RPC 服务，而要启动任何一个 RPC 服务之前，我们都需要做好 port 的对应工作
- 这个工作就是由 rpcbind 这个服务所负责的，在启动任何一个 RPC 之前，我们都需要启动 rpcbind 才行
- yum install rpcbind

2、NFS 主程序: nfs-utils

- 提供 rpc.nfsd 以及 rpc.mountd 这两个 NFS daemons 与其他相关 documents 与说明文件、可执行文件等的软件
- yum install nfs-utils

13.2.2. NFS 的软件结构

1、配置文件只有一个，执行文件也不多，记录文件也就两三个

2、主要配置文件: /etc(exports

- 系统并没有默认值，因此不一定存在
- 不存在的话手动建立即可

3、NFS 文件系统维护命令: /usr/sbin/exportfs

- 维护 NFS 共享资源的命令，可以利用这个命令重新共享/etc/export 更新的目录资源，将 NFS Server 共享的目录卸载或重新共享

4、共享资源的日志文件: /var/lib/nfs/*tab

- 在 NFS 服务器中，日志文件都放置到/var/lib/nfs/目录中
- 该目录中有两个比较重要的日志文件
 - 一个是 etab，主要记录了 NFS 所共享出来的目录的完整权限设置值
 - 另一个是 xtab，记录了曾经链接到此 NFS 服务器的相关客户端的数据

5、客户端查询服务器共享资源的命令: /usr/sbin/showmount

- 另一个 NFS 重要的命令
- exportfs 用在 NFS Server 端
- showmount 用在 Client 端

13.2.3. /etc(exports) 配置文件的语法与参数

1、NFS 会直接使用到内核的功能，所以内核必须支持 NFS 才行，通常 CentOS 或者其他发行版，默认内核通常是支持 NFS 功能的

2、NFS 服务器的搭建很简单：只要编辑好主要配置文件/etc(exports) 之后，先启动 rpcbind，然后再启动 nfs，NFS 就成功了

3、某些 distribution 并不会主动提供/etc(exports) 文件，所以需要手动建立

4、该配置文件格式

- [共享目录] [主机 1(权限)] [主机 2(权限)]
 - 共享目录：以目录为单位，这个目录可以依照不同的权限共享给不同的主机
 - 主机后面紧跟小括号"()"定义权限参数，若权限参数不止一个，则以逗号","分开
 - 主机名的设置
 - 可以使用完整的 IP 或者是网络号
 - 可以使用主机名，但主机名必须要在/etc/hosts 或者可用 DNS 找到，即可以对应到一个 IP，可以支持通配符(只有主机名支持通配符而 IP 不支持)
 - 可用权限
 - rw/ro：可读写/可读，最终是否能读写还得依照文件系统的 rwx 及身份有关
 - sync/async：sync 代表数据会同步写入到内存与硬盘中，async 则代表数据会暂存于内存当中
 - no_root_squash:
 - 默认情况下，客户端 root 身份会由 root_squash 的设置压缩成 nfsnobody，如此对服务器的系统会较有保障
 - 如果想要开放客户端使用 root 身份来操作服务器的文件系统，那么这里就需要开放 no_root_squash 才行
 - all_squash：不论登录 NFS 的用户身份为何，都会被压缩成匿名用户，通常也就是 nfsnobody
 - anonuid/anongid:
 - anon 指 anonymous(匿名用户)前面关于*_squash 提到的匿名用户的 UID 的设置值，通常为 nfsnobody
 - 你可以设置这个 UID 值，前提是这个 UID 存在于/etc/passwd 当中
 - 对应的，anongid 则是组的 GID

5、给定/etc(exports) 的配置情况，分析几种情况的访问权限

➤ 配置文件内容如下

```
/tmp      *(rw,no_root_squash)
/home/public    192.168.100.0/24(rw)      *(ro)
/home/test     192.168.100.10(rw)
/home/linux     *.centos.vbird(rw,all_squash,anonuid=45,anongid=45)
```

➤ 客户端与服务器具有相同的 UID 与账号

- 假设：我们在 192.168.100.10 登录 NFS(192.168.100.254) 服务器，并且在 192.168.100.10 的账号为 dmtsai，同时 NFS 服务器上也有 dmtsai 账号，并且具有相同的 UID

- 由于 192.168.100.254 这台 NFS 服务器的/tmp 权限为-rwxrwxrwx，所以客户端(dmtsai 在 192.168.100.10 上面)在/tmp 下面具有的访问权限，并且写入文件的属主为 dmtsai
- 在/home/public 当中，由于我们有读写的权限，所以如果/home/public 这个目录的权限对于 dmtsai 为开放写入的话，那么我们就可以读写，并且写入文件的属主是 dmtsai，但是若 NFS 服务器的/home/public 对于 dmtsai 这个用户并没有开放可写入的权限，那么我们还是没法写入文件
- 在/home/test 当中，权限与/home/public 相同，还需要 NFS 服务器的 /home/test 对于 dmtsai 的开放权限
- 在/home/linux 中，无论你是何种用户，你的身份一定会变成 UID=45 这个账号，所以这个目录就必须针对 UID=45 的那个账号名称修改它的权限才行

➤ **客户端与服务器端的账号并不相同**

- 假设：我们在 192.168.100.10 的身份为 vbird(UID=600)，但是 192.168.200.254 这台 NFS 主机却没有 UID=600 的账号
- 我们在/tmp 下面还是可以写入，只是该文件的权限会保持为 **UID=600**，因此服务器端看起来就比较奇怪，**因为找不到 UID=600 的账号，故文件属主会填上 600**
- 我们在/home/public 里面是否可以写入，还需要视/home/public 的权限而定，不过由于没有加上 all_squash 的参数，因此在该目录下会保留客户端的用户 UID
- /home/test 的权限与/home/public 相同
- /home/linux 下面，身份就变成 UID=45 这个用户

➤ **客户端的身份为 root**

- 假设：我们在 192.168.100.10 的身份为 root
- 我们在/tmp 里面可以写入，由于 no_root_squash 的参数改变了默认的 root_squash 设置值，所以在/tmp 写入的文件属主为 root，所以在/tmp 写入的文件属主为 root
- 我们在/home/public 下面的身份被设置成 nobody，因为默认属性里面有 root_squash，因此文件属主变为 nobody
- /home/test 与/home/public 相同
- /home/linux，root 的身份被压缩成 UID=45 这个用户了

6、关于文件的属主

- **关键因素是 UID，而非 UID 对应的用户名**，假设 NFS Server 与 NFS Client 上有相同 UID 的账号，但是用户名不同(分别为 UserServer, UserClient)，那么 NFS Client 在共享目录上创建的文件(在 Client 端看来，属主是 UserClient)，在 NFS Server 端看来属主就是(UserServer)

13.2.4. 启动 NFS

1、启动命令

- systemctl restart rpcbind.service
- systemctl nfs.service

2、rpcbind 启动的 port 在 111，同时启动在 UDP 与 TCP

3、查询每个 RPC 服务器的注册情况<rpcinfo>

- rpcinfo -p [IP|hostname]

- `rpcinfo -t|-u IP|hostname` 程序名称
- `-p:` 针对某 IP(未写则默认为本机)显示出所有 port 与 program 的信息
- `-t:` 针对某主机的某个程序检查其 TCP 数据包所在的软件版本
- `-u:` 针对某主机的某支程序检查其 UDP 数据包所在的软件版本

13.2.5. NFS 的连接查看

1、在 NFS 服务器设置妥当之后，我们可以在 Server 端先自我测试一下是否可以连接，利用`<showmount>`

- `showmount [-ae] [hostname|IP]`
- `-a:` 显示当前主机与客户端的 NFS 连接共享的状态
- `-e:` 显示某台主机的/etc/exports 所共享的目录数据
- `showmount -e hocalhost` <==在服务器端查询
- `showmount -e < NFS 服务器 IP 或主机名 >` <==在客户端查询

2、/etc/exports 只是比较特别的权限参数而已，还有很多默认参数，可以检查文件/**var/lib/nfs/etab**

3、如果有其他客户端挂在了你的 NFS 文件系统，那么该客户端与文件系统信息就会被记录到/var/lib/nfs/xtab 中去

4、如果想要重新处理/etc/exports 文件，当重新设置完/etc/exports 后需要重新启动 NFS 吗？不需要，可以用`<exportfs>`来帮忙

- `exportfs [-aruv]`
- `-a:` 全部挂载(或卸载) /etc/exports 文件中的设置
- `-r:` 重新挂载 /etc/exports 里面的设置，此外，亦同步更新/etc/exports 以及/var/lib/nfs/xtab 的内容
- `-u:` 卸载某一目录
- `-v:` 在 export 的时候，将共享的目录显示到屏幕上

13.2.6. NFS 的安全性

1、防火墙的设置问题与解决方案

- 一般来说，NFS 的服务仅仅对内部网络开放，不会对因特网开放
- NFS 的防火墙特别难设置，因为除了固定的 port 111 与 port 2049 之外，**还有很多由 rpc.mountd、rpc.rquotad 等服务所开启的不固定的端口，所以 iptables 就很难设定规则**
- 为了解决这个问题，CentOS 6.x 开始提供了一个固定特定 NFS 服务的端口配置文件(/etc/sysconfig/nfs)，在这个文件里就能能够指定特定的端口，这样每次启动 NFS 时，相关服务启动的端口就会固定，这样就可以设置正确的防火墙了
- 与 NFS 相关的主要 RPC 服务有：mountd、rquotad、nlockmgr
- `systemctl restart nfs.service`
- `rpcinfo -p | grep -E '(rquota|mount|nlock)'` <==<未完成>：好像并未读取修改的配置文件

2、使用/etc/exports 设置更安全的权限

- 善用 root_squash 以及 all_squash 等功能，在利用 anonuid 等设置来规范登录主机的用户身份

- NFS 服务器的文件系统权限设置不要随便设定称为 `rwxrwxrwx`, 这样会給系统留下非常大的安全隐患

3、更安全的 partition 规划

- 如果工作环境中有多台 Linux 主机, 并且打算彼此共享目录, 那么在安装 Linux 的时候, 最好规划处一块 partition 作为预留之用, 因为 NFS 可以針對目录来共享
- 可以将预留的 partition 挂载在任何一个挂载点, 再将挂载点(就是目录)在 `/etc/exports` 的设置共享出去, 那么整个工作环境中的其他 Linux 主机就可以使用该 NFS 服务器的那块预留 partition 了
- 在这种设置下, 只需要留意该 partition 而已

4、NFS 服务器关机前的注意事项

- 当 NFS 使用这个 RPC 服务在客户端连上服务时, 那么服务器想要关机, 可能会成为不可能的任务, 如果服务器上面还有客户端在连接, 那么可能需要等几个小时才能正常关机
- 建议 NFS Server 在想要关机之前, 先关掉 `rpcbind` 与 `nfs` 这两个 daemon
- 或者利用 `showmount -a localhost` 查找出哪个客户端还在连接, 或者是通过查阅 `/var/lib/nfs/rmtab` 或 `xtab` 等文件来检查亦可, 找到这些客户端后, 直接 call 它们, 让它们帮忙先挂断服务

13.3. NFS 客户端的设置

13.3.1. 手动挂载 NFS 服务器共享的资源

1、步骤

- 确认本地端已经启动了 `rpcbind` 服务
- 扫描 NFS 服务器共享的目录有哪些, 并了解我们是否可以使用(`showmount`)
- 在本地端建立预计要挂载的挂载点目录(`mkdir`)
- 利用 `mount` 将远程主机直接挂载到相关目录

2、详细操作

- `showmount -e 192.168.200.254` <=> 在 Client 端查看 NFS 服务器可共享的文件有哪些
- `mkdir -p <挂载点目录>`
- `mount -t nfs 192.168.200.254:<共享目录> <本地挂载点>`
- `umount <本地挂载点>`

13.3.2. 客户端可处理的挂载参数与开机挂载

1、问题引入

- 如果 NFS 服务器共享目录内有个 `script` 的内容为 `rm -rf /`, 文件权限为 555, 如果你不小心执行了这个脚本, 那么客户端整个系统都坏了
- 因此除了 NFS 服务器需要保护之外, 使用人家的 NFS 文件系统也需要自我保护, 可以通过 `mount` 的命令参数来实现

2、`<mount>`主要参数

- `mount [-t 文件系统] [-o 额外选项] [设备文件名] [挂载点]`
- 以下为额外选项说明, 标记绿色的为系统默认值
 - `suid, nosuid`: 是否允许此分区含有 SUID/SGID 的文件格式, 当挂载的

`partition` 上面有任何 SUID 的 binary 程序时，只要使用 `nosuid` 就能取消 SUID 的功能

- `ro, rw`: 挂载文件系统成为只读(`ro`)或可读写(`rw`)。服务器可以提供可读写，但是客户端仅可允许只读的参数设置值
- `dev, nodev`: 是否允许此分区可以创建设备文件，一般来说只有`/dev`这个目录才会有特殊的设备
- `exec, noexec`: 是否允许此分区上拥有可执行的 `binary` 文件
- `user, nouser`: 是否允许此分区让任何用户执行 `mount` 和 `umount` 操作，一般来说 `mount` 只有 `root` 可以进行，如果要保护好文件系统，最好不要允许用户进行挂载与卸载
- `auto, noauto`: 是否允许此分区被以 `mount -a` 自动挂载
- `async, sync`: 文件系统是否使用同步写入(`sync`)，或异步(`async`)的内存机制，默认为 `async 3`

3、关于 NFS 特殊的挂载参数

- 当我们进行任何操作时，只要是用到文件系统，那么整个目录树系统就回去主动查询全部的挂载点，如果 NFS 服务器与客户端之间的连接因为网络问题，或者是服务器端先关机了却没有通知客户端，那么客户端只要使用到文件系统的命令(如 `df ls cp` 等)，整个系统就会变得非常慢，因为必须要等到文件系统查找等待超时后，才能继续工作
- 为了避免上述问题，提供了一些额外的 NFS 挂载参数，绿色标记的为系统默认值
 - `fg, bg`: 当执行挂载时，该挂载的行为会在前台(`fg`)还是在后台(`bg`)执行
 - 若在前台执行，则 `mount` 会持续尝试挂载，直到成功或 `time out` 为止
 - 若为后台执行，则 `mount` 会在后台持续多次进行 `mount`，而不会影响到前台程序的运行
 - `soft, hard`:
 - 若是 `hard`: 则两者之间有任何一台主机脱机，RPC 会持续呼叫，直到对方恢复连接为止
 - 若是 `soft`: RPC 会在 `time out` 后重复呼叫，而非持续呼叫，因此系统的延迟将不是特别明显
 - 若服务器可能开开关关，建议 `soft`
 - `intr`: 当使用 `hard` 方式挂载时，若加上 `intr` 这个参数，当 RPC 持续呼叫时，该次呼叫是可被中断的，**无默认值**
 - `rsize, wsize`: 读出(`rsize`)与写入(`wsize`)区块大小，这个设置值可以影响客户端与服务器端传输数据的缓冲记忆容量
 - 一般来说，如果在局域网内，并且客户端与服务器端都具有足够的内存，那这个值可以设置的大一点，提升缓冲区大小将提升 NFS 文件系统的传输能力，但不要设置得太大
 - **`rsize=1024 <==默认大小`**
 - **`wsize=1024 <==默认大小`**

4、使 NFS 开机即挂载

- 开机挂载的挂载点与相关参数是写入`/etc/fstab` 文件的
- NFS 不能写入该文件(`/etc/fstab`): 因为网络的启动时在本机挂载之后
 - 当你利用`/etc/fstab` 尝试挂载 NFS 时，系统由于尚未启动网络，因此肯定

- 无法挂载成功的
➤ 写入/etc/rc.d/rc.local 即可

13.3.3. 无法挂载的原因分析

- 1、客户端的主机名或 IP 网段不被允许使用
- 2、服务器或客户端某些服务未启动
- 3、被防火墙拦截
- 4、无法卸载的原因以及解决办法<fuser>
 - device is busy <==可能你当前就在该目录下，所以无法卸载
 - fuser -m -v [挂载点] <==查询哪些程序在占用这个目录
 - fuser -m -k -i [挂载点] <==杀死哪些占用这个目录的程序，-i 参数代表杀死改程序前询问你

13.3.4. 自动挂载 autofs 的使用

- 1、在一般 NFS 文件系统的使用过程中，如果客户端要使用服务器端所提供的 NFS 文件系统，要么就是得在/etc/rc.d/rc.local 中设置开机时挂载，要么就登录系统后手动利用 mount 挂载。此外，客户端需要预先手动建立好挂载点目录，然后挂载上来
- 2、NFS 文件系统与网络连接的困扰
 - 如果挂载了 NFS 服务器后，任何一方脱机都可能造成另外一方总是在等待超时
 - 挂载的 NFS 文件系统又不是常常被使用，但若不挂载的话，有时候紧急要使用时又得通知系统管理员，这很不方便
 - 换一个角度思考问题
 - 可不可以让客户端在使用 NFS 文件系统的需求时才让系统自动挂载
 - 当 NFS 文件系统使用完毕后，可不可以让 NFS 自动卸载，以避免可能的 RPC 错误
- 3、autofs 的配置概念
 - autofs 这个服务在客户端计算机上面，会持续检测某个制定目录，并预先设置当使用到该目录下的某个子目录时，将会取得来自服务器端的 NFS 文件系统资源，并进行自动挂载的操作
 - autofs 的主要配置文件为/etc/auto.master
 - 这个文件内容很简单，只要定义出最上层目录(/etc/nfsfile)即可
 - 这个目录就是 autofs 会一直持续检测的目录
 - 后面跟的文件(/etc/auto.nfs,名字可以自定义)则是与该目录下各自目录相对应，该文件/etc/auto.nfs 可以定义出每个子目录所欲挂载的远程服务器的 NFS 目录资源
- 4、建立主配置文件/etc/auto.master
 - vim /etc/auto.master
 - /home/nfsfile /etc/auto.nfs
 - 注意/home/nfsfile 目录不需要事先存在，因为 autofs 会主动建立该目录，如果你建立，反而有可能出问题
- 5、建立数据对应文件内(/etc/auto.nfs)的挂载信息与服务器对应的资源
 - 该文件是自己设置的，名字任取，我们需要自己创建

- vim /etc/auto.nfs
 - linux -rw, bg, soft, rsize=32768, wsize=32768 192.168.200.254:/home/linux
 - tmp -rw, bg, soft, rsize=32768, wsize=32768 192.168.200.254:/home/tmp
- 这里指定的 linux 与 tmp 目录都是在/home/nfsfile 为顶层目录下的相对路径，因此为/home/nfsfile/linux 与/home/nfsfile/tmp

13.4. 案例演练

1、<未完成>

Chapter 14. 账号管理：NIS 服务器

14.1. NIS 的由来与功能

1、在一个大型网络中，如果有多台 Linux 主机，每一台主机都要设置相同的账号与密码，如果有一台账号主控服务器来管理网络中所有主机的账号，当其他的主机有用户登录的需求时，才到这台主控服务器上面要求验证相关的账号、密码等用户信息，如此一来，如果想要增加，修改，删除用户数据，只要这台主控服务器上面处理即可

14.1.1. NIS 的主要功能：管理账号信息

1、通常我们会建议，一台 Linux 主机的功能越单一越好，也就是说，一台 Linux 就专门进行一项服务

- 由于功能单一所以系统资源可以被完整运用，并在发生入侵或者是系统产生问题的时候，也比较容易追查问题所在

2、这样虽然有分散风险、容易追踪的好处，但是由于同一个公司内的多台主机，所以事实上所有的 Linux 主机的账号与密码都是一样的

- 如果设计了一台专门管理账号与密码的服务器，而其他的 Linux 主机当有客户端要登录的需求时，就必须到这台管理密码的服务器来查询用户的账号与密码
- 如此一来，要管理所有的 Linux 主机的账号与密码，只要到那台主机的服务器上面去进行设置即可
- 这就是 Network Information Service

3、**Network Information Service** 最早应该称为 Sun Yellow Pages(简称 YP)，也就是 Sun 公司出的一个名为 Yellow Pages 的服务器软件，NIS 与 YP 是一模一样的东西

4、NIS 服务器提供的数据

- /etc/passwd: 提供账号、UID、GID、用户主目录位置、Shell 等
- /etc/group: 提供组数据以及 GID 的对应，还有该组的成员用户
- /etc/hosts: 主机名与 IP 的对应，常用于 private IP 的主机名对应
- /etc/services: 每一种服务(daemons)对应的端口(port number)
- /etc/protocols: 基础的 TCP/IP 数据包协议，如 TCP、UDP、ICMP 等
- /etc/rpc: 每种 RPC 服务器所对应的程序号码
- /var/yp/ypservers: NIS 服务器所提供的数据库

14.1.2. NIS 的工作流程：通过 RPC 服务

1、由于 NIS 服务器主要是提供给用户的登录信息给客户端主机来查询之用，**所以 NIS 服务器所提供的数据当然就要用到传输与读写比较快的数据库文件系统，而不是传统的纯文本数据**

- 为了达到这个目的，NIS 服务器必须将前一小节提到的哪些文件制作称为数据库文件，然后使用网络协议让客户端主机来查询
- 使用的通信协议与 NFS 相同，都是远程过程调用(RPC)这个协议

2、较为大型的企业环境中，NIS 服务器可以使用 Master/Slave(主控/辅助服务器)架构来避免一台 NIS 服务器宕机导致整个网络无法登陆的情况

- Master NIS 服务器提供系统管理者制作的数据库
- Slave 则复制来自 Master 的数据，进而向其他客户端提供查询
- 客户端可以向整个网络请求用户资料的响应，Master 与 Slave 皆可应答，由于 Slave 的数据来自于 Master，所以用户账号数据本身是同步的，这样可以分散 NIS 服务器的负载

3、首先必须要有 NIS Server 存在，之后才会有 NIS Client 的存在，当用户有登录需求时，整个 NIS 运作过程如下

- 关于 NIS Server(Master/Slave)的工作流程
 - NIS Master 先将本身的账号密码相关文件制作称为数据库文件
 - NIS Master 可以主动告知 NIS Slave Server 进行更新
 - NIS slave 也可主动前往 NIS Master Server 取得更新后的数据库文件
 - 若出现账号密码的变动时，需要重新制作 database 并重新同步 Master/Slave
- 关于当 NIS Client 有任何登录查询的需求时
 - NIS Client 若有登录需求时，会先查询其本机的/etc/passwd、/etc/shadow 等文件
 - 若在 NIS Client 本机找不到相关的账号数据，才开向整个 NIS 网络的主机广播查询
 - 每个 NIS Server(不论 Master/Slave)都可以相应，基本上"先响应者优先"

4、如果 NIS Client 本身就有很多普通用户账号时，那跟 NIS Server 所提供的账号就可能产生一定程度的差异

- 一般来说，在这样的环境下，NIS Client 或 NIS Slave Server 会主动拿掉自己本机的普通用户账号，仅保留系统所需要的 root 及系统账号
- 这样，一般用户才会通过 NIS Master Server 进行控制

5、**NIS 环境大致上需要设置的基本组件：**

- NIS Master Server：将文件转换为数据库，并提供 Slave Server 来更新
- NIS Slave Server：以 Master Server 的数据库作为自身的数据库来源
- NIS Client：向 Master/Server 请求进行登录用户身份验证

14. 2. NIS Server 端的设置

14. 2. 1. 所需要的软件

1、由于 NIS 服务器需要使用 RPC 协议，且 NIS 服务器同时也可以被当成客户端，因此它需要的软件就有下面这几个

- yp-tools：提供 NIS 相关的查询命令功能
- ypbind：提供 NIS Client 端的设置软件
- ypserv：提供 NIS Server 端的设置软件
- rpcbind：这是 RPC 必须的软件

14. 2. 2. NIS 服务器相关的配置文件

1、NIS 服务器上最重要的就是 ypserv 这个软件

2、由于 NIS 设置时还会使用到其他网络参数，因此在配置文件方面需要有下面这些数据

- /etc/ypserv.conf：最主要的 ypserv 软件所提供的配置文件，可以定义 NIS

客户端是否有可登陆的权限

- `/etc/hosts`: 由于 NIS Server/Client 会用到网络主机名与 IP 对应, 因此这个主机名对应文件比较重要, 每一台主机名与 IP 都需要记录才行
- `/etc/sysconfig/network`: 可以在这个文件指定 NIS 的网络
- `/var/yp/Makefile`: 与监理数据库有关的操作控制文件

3、NIS 服务器提供的主要服务有以下两个

- `/usr/sbin/ypserv`: NIS 服务器主要提供的服务
- `/usr/sbin/rpc.yppasswdd`: 提供额外的 NIS 客户端的用户密码修改服务, 通过这个服务, NIS 客户端可以直接修改在 NIS 服务器上的密码, 相关的使用程序时 `yppasswd`

4、与账号密码的数据库有关的命令有以下几个

- `/usr/lib64/yp/ypinit`: 建立数据库的命令, 经常使用(32 位平台下是 `/usr/lib/yp/ypinit`)
- `/usr/bin/yppasswd`: 与 NIS 客户端有关, 主要让用户修改服务器上的密码

14.2.3. 一个实际操作案例

1、NIS 的域名为

2、整个内部的信任网络为: 192.168.200.0/24

3、NIS Master Server 的 IP 为 192.168.200.254, 主机名为

4、NIS Client 的 IP 为 192.168.200.101, 主机名为

14.2.4. NIS Server 的设置与启动

1、先设置 NIS 的域名

- `vim /etc/sysconfig/network`
NISDOMAIN=liuyenis
`YPSERV_ARGS="-p 1011"`

2、主要配置文件`/etc/ypserv.conf`

3、设置主机名与 IP 的对应(`/etc/hosts`)

- NIS 大部分是给局域网内的主机使用的, 所以不需要 DNS 的设置
- 由于 NIS 使用到很多的主机名, 而且网络连接时利用 IP 进行的, 因此一定要设置好`/etc/hosts`里面的主机名与 IP 的对应, 否则会无法成功连接 NIS
- 如果主机名与 NIS 的主机名不一样, 那么在这个文件当中还需要将你的主机名设置进来, 否则在后面数据库的设置时, 会发生问题

4、启动与查看所有相关服务

- `vim /etc/sysconfig/yppasswdd`
`YPPASSWDD_ARGS="--port 1012"`
- `systemctl restart ypserv.service`
- `systemctl restart yppasswdd.service`
- `rpcinfo -p localhost`
- `rpcinfo -u localhost ypserv`

5、处理账号并建立数据库

- `useradd -u 1001 nisuser1`
- `useradd -u 1001 nisuser2`

- useradd -u 1001 nisuser3
- echo password | passwd --stdin nisuser1
- echo password | passwd --stdin nisuser2
- echo password | passwd --stdin nisuser3
- **/usr/lib64/yp/ypinit -m**
 - next host to add: <==这里按下[ctrl]+d
- 当 NIS Server 端账号有增减情况时，该数据库好像不会自动更新，再执行一次上述命令更新数据库即可

14.2.5. 防火墙设置

1、增加以下几条

- iptables -A INPUT -i \$EXTIF -p tcp -s 192.168.200.0/24 --dport 1011 -j ACCEPT
- iptables -A INPUT -i \$EXTIF -p udp -s 192.168.200.0/24 -m multiport --dport 1011,1012 -j ACCEPT

2、我实际上已经对局域网开放所有的端口了，不需要任何设置了

14.3. NIS Client 端的设置

1、网络连接是双向的，所以 NIS Server 提供数据库文件外，NIS Client 也需要提供连接软件，即 rpcbind

2、当 NIS Client 端有登录需求时，NIS Client 基本上还是先查找自己的/etc/passwd、/etc/group 等数据后才再去找 NIS Server 的数据库

- 因此 NIS Client 最好能够将本身的账号密码删除到仅剩下系统账号，即 UID、GID 小于 500 的账号
- 如此一来，即可让系统执行无误，也能够让登录用户的信息完全来自 NIS Server，比较单一

14.3.1. NIS Client 所需的软件与软件结构

1、NIS Client 端所需要的软件有

- ypbind: 与 ypserv 相互沟通的客户端连接软件
- yp_tools: 提供查询的软件

2、在配置 NIS Client 时，你可能需要用到下面的文件

- /etc/sysconfig/network: NIS 的域名
- /etc/hosts: 至少需要有各个 NIS 服务器的 IP 与主机名的对应
- /etc/yp.conf: 这是个 ypbind 的主要配置文件，里面主要规范 NIS 服务器
- /etc/sysconfig/authconfig: 规范账号登录时的允许认证机制
- /etc/pam.d/system-auth: 账号通常由 PAM 模块管理，所以必须要在 PAM 模块内加入 NIS 的支持才行
- /etc/nsswitch.conf: 这个文件可以规范账号密码与相关信息的查询顺序，默认是先找/etc/passwd 再找 NIS 数据库

3、NIS 客户端可以使用的命令

- /usr/bin/yppasswd: 更改你再 NIS database(NIS Server 所制作的数据库)中的密码
- /usr/bin/ypchsh: 同上，但是更改 shell

- /usr/bin/ypchfn: 同上, 更改一些用户信息

14.3.2. NIS Client 的设置与启动

1、setup

- 选择 Authentication configuration, 然后 RunTool
- 选中 Use NIS (按空格选中), 然后 next
- 写入 Domain 与 Server 的 IP, 然后 OK

2、查看是否成功

- cat /etc/sysconfig/network <==NIS Server 的配置信息会写入到这里
- cat /etc/yp.conf <==会看到 setup 中设置的 Domain 与 Server
- vim /etc/nsswitch.conf <==看到如下表明 nis 已经支持
 - passwd: files nis sss
 - shadow: files nis sss
 - group: files nis sss
 - hosts: files nis dns

3、由于改动的文件实在太多了, 建议用 setup 来设置

4、如果要手动处理的话, 需要修改以下文件

- /etc/sysconfig/network
- /etc/nsswitch.conf
- /etc/sysconfig/authconfig
- /etc/pam.d/system-auth
- /etc/yp.conf

14.3.3. NIS Client 端的验证: yptest、ypwhich、ypcat

1、利用<ypertest>验证数据库

- yptest <==出现"Test 9: yp_all"说明验证成功

2、利用<ypwhich>检查数据库数量

- ypwhich <==显示 NIS Server 的 Domian 名称
- ypwhich -x <==显示 NIS Client 与 Server 之间沟通的数据库有哪些
- 这些数据库是放在 NIS Server 的/var/yp/< NISDOMAIN >/ * 里面

3、利用<ypcat>读取数据库内容

- ypcat [-h nisserver] [数据库名称]
- -h nisserver: 如果已经设置的话, 指向某一台特定的 NIS 服务器, 如果没有指定的话, 以 ypbind 的设置为主
- 数据库名称: 即在/var/yp/< NISDOMAIN >/内的文件名, 例如 passwdbyname
- ypcat passwdbyname

14.3.4. 用户参数修改: yppasswd、ypchfn、ypchsh

1、由于 NIS Client 是通过数据库来取得用户的账号密码的, 那么如何在 NIS 客户端处理账号密码的修改呢

- 这就是为什么要在 NIS Server 中启动 yppasswdd 这个服务的主要用意
- yppasswdd 可以接受 NIS Client 端发送来的密码修改要求, 进而处理 NIS Server 的/etc/passwd、/etc/shadow, 然后 yppasswdd 还能够重建密码数据库, 让 NIS Server 同步更新数据库

- su - nisuser1
 - id
 - yppasswd

14.4. NIS 搭配 NFS 的设置在群集计算机上的应用

1、为什么上一小节登录 nisuser1 没有用户主目录?

- nisuser1 的用户主目录是在服务器端的/home 上，而在客户端登录时，在客户端的/home 目录下根本不可能有 nisuser1 的用户目录
- 可以将服务器端的/home 挂载到客户端上面即可

2、什么是群集计算机

- 在超级计算机中，主要是通过内部电路将很多 CPU 与内存连接在一块，因为是特殊设计，因此价格非常昂贵
- 如果我们可以将较便宜的个人计算机连接在一块，然后将数值运算的任务分别丢给每一台连接在一块的个人计算机，那就很像超级计算机了

3、这一做法有几个限制

- 因为每台计算机都需要运算相同的程序，而我们知道运算的数据都在内存中
- 程序启动时需要给予一个身份，并且程序读取的程序在每台计算机上面都需要相同
- 同时每台计算机都需要支持并行化运算

4、因此在 PC Cluster 上面所有计算机就需要有

- 相同的用户账户信息，包括账号、密码、用户主目录等一大堆信息
 - 可以通过 NIS 来处理
- 相同的文件系统，例如/home、/var/spool/mail 以及数值程序放置的位置
 - 可以使用 NFS 来搞定
- 可以搭配的并行化函式库，常见的 MPICH、PVM 等

5、一个 PC Cluster 的实例

- 规划
 - 账号：建立 UID 大于 2000 的账号，名称为 cluser1、cluser2、cluser3，且这些账号的用户主目录预计放置与/rhome/目录内，与 NIS Client 本地的用户分开
 - NIS 服务器：域名:liuyecluster，
服务器是:www.centos.liuye(192.168.200.254)
 - NIS 客户端：192.168.200.{101~200}(dpch)
 - NFS 服务器：服务器共享了/rhome 给 192.168.200.0/24 这个网络，且预计所有程序放置于 cluster 目录中，此外假设所有客户端都是很干净的系统，不需要压缩客户端的身份(NFS 的设置)
 - NFS 客户端
- NIS 配置阶段
 - **NIS Server 端的操作**
 - mkdir /rhome
 - useradd -u 2001 -d /rhome/cluser1 cluser1
 - useradd -u 2002 -d /rhome/cluser2 cluser2

- useradd -u 2003 -d /rhome/cluser3 cluser3
- echo password | passwd --stdin cluser1
- echo password | passwd --stdin cluser2
- echo password | passwd --stdin cluser3
- vim /etc/sysconfig/network
NISDOMAIN=liuyecluster
- **nisdomainname liuyecluster**
- **systemctl restart ypserv.service**
- **systemctl yppasswd.service**
- **/usr/lib64/yp/ypinit -m**
- 以上 4 个操作有依赖性， 请顺序执行

- **NIS Client 端的操作**

- 以 setup 进行 NIS 的设置
- id cluser1 确认

- NFS 服务器的设置

- NFS Server 端的操作

- mkdir /cluster
- vim /etc/exports
 - /rhome 192.168.200.0/24(rw,no_root-squash)
 - /cluster 192.168.200.0/24(rw,no_root_squash)
- systemctl restart nfs.service
- showmount -e localhost

- NFS Client 端的操作

- mkdir /rhome /cluster
- mount -t nfs 192.168.200.254:/rhome /rhome
- mount -t nfs 192.168.200.254:/cluster /cluster

- 至此， 大功告成

Chapter 15. 时间服务器：NTP 服务器

- 1、计算机内部所记录的时钟是记载于 BIOS(CMOS)内的，但如果计算机上面的 CMOS 电池没电了，或者某些特殊因素导致 BIOS 数据被清除，此时计算机的时间就会不准，同时由于某些操作系统程序的问题，也可能出现我们看到的事件与现实社会不相同的情况
- 2、因此，我们都会调整一下事件，以便让计算机系统的时间可以一直保持正确的状态

15.1. 关于时区与网络校时的通信协议

1、每一台主机的时间同步化很重要

- 如果搭建了一台日志文件服务器的话，那么总需要分析每台主机所传来的日志文件信息，如果每一台主机的事件都不相同，那就没办法判断问题发生的时刻了
- DHCP 客户端/服务器端所需要的租约事件限制
- 网络监测时所需要注意的时间点
- 群集计算机群等

15.1.1. 什么是时区？全球有多少时区？GMT 在哪个时区

1、地球分成了 24 个时区

- 经度的零点在英国格林尼治这个城市所在的纵剖面上(纵剖面就是由南极切到北极的直线，横切面就是与赤道平行的切线)
- 格林尼治时间(Greenwich Mean Time,GMT 时间)为标准时间
- 北京时间是 GMT+8(北京比 GMT 快了 8 小时)
- 日期变更线：
 - 由于格林尼治东边时间较早，西边时间较晚
 - 但是两边各走 180 度后就会碰头，就刚好差了 24 小时

15.1.2. 什么是夏令时(Daylight Saving Time)

1、夏令时(日光节约时间)

- 在夏天的时候，白天的时间会比较长，为了节约用电，夏天的时候某些地区会将时间提早一小时，也就是说，原本的时区是 8 点，因为夏天太阳升起的时间比较早，因此把事件向前挪，在原本 8 点的时候，定为该天的 9 点
- 如此一来，可以利用阳光照明，省去了花费电力的时间，因此才会称为夏令时

15.1.3. Coordinated Universal Time(UTC) 与时间系统的误差

1、以下要谈的是正确的时间

2、1880 年的时间标准是以 GMT 时间为主的，但是 GMT 时间是以太阳通过格林尼治那一刻来作为计时的标准

- 然而地球自转的轨道以及公转的轨道并非正圆，加上地球的自转速度有逐年递减的问题，所以这个时候 GMT 时间与我们目前计时的时间就有点不

一样了

3、最准确的应该是使用**原子振荡周期**所计算的物理时钟(Atomic Clock, 也称为原子钟)

- 我们常看到的 UTC, 即 Coordinated Universal Time(协和标准时间), 就是利用这种 Atomic Clock 为基准所定义出来的正确时间

4、BIOS 内部就含有一个原子钟在记录与计算时间的进行

- 主要利用计算芯片(crystal)的原子振荡周期来计时的, 这是因为每种芯片都有自己的独特的振荡周期
- 但这种芯片的振荡周期多少有些差异

15.1.4. NTP 通信协议

1、Linux 操作系统的计时方式主要从 1970 年 1 月 1 日开始计算总秒数

- date 有个%s 参数, 可以取得总秒数, 这个就是软件时钟
- 计算机硬件主要是以 BIOS 内部的时间为主要的时间依据(硬件时钟)
- 软件时钟: 由 Linux 操作系统根据 1970/1/1 开始计算的总秒数
- 硬件时钟: 主机硬件系统上面的时钟, 例如 BIOS 记录的时间

2、如果选择几部主要主机(Primary Server)调校事件, 让这些 Primary Server 的时间同步之后, 再开放网络服务来让 Client 端连接, 并且允许 Client 端调整自己的时间

3、NTP(Network Time Protocol)

- 首先, 主机要启动 daemon
- 之后, Client 会向 NTP Server 发出校对时间的 message
- 然后 NTP Server 会送出当前的标准时间给 Client
- Client 接收了来自 Server 的时间后, 会据以调整自己的时间, 这样就实现了网络校对

4、如果 Client 到 Server 的信息传送时间过长怎么办

- 一些 program 已经开发了自动计算时间传送过程的误差, 以更准确地校准自己的事件
- 在 daemon 部分, 也同时提供以 Server/Client 以及 Master/Slave 的架构来提供用户进行网络校时的操作
 - 我国的标准时间主机去国际标准时间的主机校时
 - 各大院校再到我国的标准时间主机校时
 - 我们再到各大院校的标准时间主机校时

5、NTP 这个 daemon 以 port 123 位连接的端口(UDP 数据包)

- 使用 NTP 软件提供的 ntpdate 来进行 port 123 的连接

15.1.5. NTP 服务器的层次概念

1、NTP 时间服务器采用类似分级架构(stratum)来处理时间的同步化

- 使用的是类似一般 Server/Client 的主从架构
- 网络社会上提供一些主要与次要的时间服务器, 这些均属于第一级与第二级的时间服务器

2、如果确实有假设 NTP 的需求时, 我们可以直接选择我国的上层 NTP 来同步化时间即可(210.72.145.44)

- 一般来说, 我们会选择多台上层的 Time Server 来作为我们这一台 NTP

Server 的校时之用，选择多台的原因是因为可以避免某台时间服务器突然宕机，其他主机仍然可以提供 NTP 主机来自我更新，然后 NTP Server 才提供给自己的 Client 端更新时间

15.2. NTP 服务器的安装与设置

1、你要做的就是将它安装起来之后，定义一台上层 NTP 服务器来同步化你的时间即可，如果你只是要进行你自己单部主机的时间同步化，那就别假设 NTP，直接使用 NTP 客户端软件即可

15.2.1. 所需软件与软件结构

1、与时区相关的数据文件所需要的软件

- `ntp`: 就是 NTP 服务器的主要软件，包括配置文件以及可执行文件
- `tzdata`: Time Zone Data 的缩写，提供各时区对应的显示格式

2、与时间以及 NTP 服务器设置相关的配置文件与重要数据文件有下面几个

- `/etc/ntp.conf`: 就是 NTP 服务器的主要配置文件，也是唯一的一个
- `/usr/share/zoneinfo/`: 由 `tzdata` 所提供，为各时区的事件格式对应文件
 - 例如我国时区格式对应文件是`/usr/share/zoneinfo/Asia/Shanghai`
- `/etc/sysconfig/clock`: <没找到>设置时区与是否使用 UTC 时钟的配置文件，每次开机后，Linux 会自动读取这个文件来设置自己系统所默认要显示的时间
- `/etc/localtime`: 本地端的时间配置文件，Linux 系统会将 Shanghai 那个文件复制一份成为`/etc/localtime`

3、常用于事件服务器与修改时间命令

- `/bin/date`: 用于 Linux 事件的修改与显示的命令
- `/sbin/hwclock`: 用于 BIOS 时钟的修改与显示的命令
 - root 才能执行，也就是说 Linux 系统上面的 BIOS 时间与 Linux 系统时间是分开的
 - 所以使用 `date` 调整时间之后，还需要使用 `hwclock` 才能将修改后的时间写入 BIOS 中
- `/usr/sbin/ntpdate`: 主要提供 NTP 服务的程序，配置文件为`/etc/ntp.conf`
- `/usr/sbin/ntpdate`: 用户客户端的时间校正，如果不要启用 NTP 而仅想使用 NTP Client 功能的话，会用到这个命令

15.2.2. 主要配置文件 `ntp.conf` 的处理

1、NTP 服务器所需要的架构如下

- 上层 NTP 服务器(`s2c.time.edu.cn: 202.112.10.36`)
- 不对 Internet 提供服务，仅允许来自内部网络 `192.168.200.0/24` 的查询
- 检测 BIOS 时钟与 Linux 系统时间的差异并写入`/var/lib/ntp/drift` 文件中

2、利用 `restrict` 来管理权限控制

- 在 `ntp.conf` 文件内可以利用 `restrict` 来控制权限，该参数设置方式如下
`restrict [你的 IP] mask [netmask_IP] [parameter]`
- `parameter` 的参数：
 - `ignore`: 拒绝所有类型的 NTP 连接

- **nomodify**: 客户端不能使用 ntpc 与 ntpq 这两个程序来修改服务器的时间参数，但客户端仍可通过这部主机来进行网络校时
- **noquery**: 客户端不能使用 ntpq、ntpc 等命令来查询时间服务器，等于不提供 NTP 的网络校时
- **notrap**: 不提供 trap 这个远程事件登录(remote event logging)的功能
- **notrust**: 拒绝没有认证的客户端
- **如果没有在 parameter 的地方加上任何参数，表示：该 IP 或网段不受任何限制**

➤ 一般来说，我们可以先关闭 NTP 的权限，然后再一个个地启用允许登录的网段

➤ **记得放行上层 NTP 服务器**

3、利用 server 设置上层 NTP 服务器

➤ 上层 NTP 服务器的设置方式如下

server [IP or hostname] [prefer]

➤ prefer 表示优先使用的服务器

4、以 driftfile 记录时间差异

➤ 设置的方式如下

driftfile [可被 ntpd 写入的目录与文件]

➤ 因为默认的 NTP Server 本身的事件计算是依据 BIOS 的芯片振荡周期频率来计算的，但是这个数值与上层 Time Server 不见得一致，所以 NTP 这个 daemon(ntpd)会自动去计算我们自己主机的频率与上层 Time Server 的频率，并将两个频率的误差记录下来，记录下来的文件就是 driftfile 后面所指的文件

➤ 关于文件的注意点

- driftfile 后面接的文件需要使用完整路径文件名
- 该文件不能是连接文件
- 该文件要设置成 ntpd 这个 daemon 可以写入的权限
- 该文件所记录的数值单位为百万分之一秒(ppm)

➤ 该文件会被 ntpd 自动更新，因此权限一定要让 ntpd 可写入才行

5、keys[key_file]

➤ 除了以 restrict 来限制客户端的连接之外，可以通过密钥系统给客户端认证

15.2.3. NTP 的启动与观察

1、systemctl restart ntpd.service

2、<ntpstat>: 查看我们的 NTP 服务器是否已经顺利地更新了时间

3、<ntpq>

➤ ntpq -p <=列出当前我们 NTP 与相关的上层 NTP 的状态

➤ 输出字段意义解释

- **remote**: NTP 主机或 IP 或主机名
 - 如果有*代表目前正在作用当中的上层 NTP
 - 如果是+代表连接成功，而且可作为下一个提供事件更新的候选者
- **refid**: 参考的上一层 NTP 主机的地址
- **st**: 就是 stratum 阶层

- **when:** 几秒钟前曾经做过时间同步化更新的操作
- **poll:** 下一次更新再几秒钟之后
- **reach:** 已经向上层 NTP 服务器要求更新的次数
- **delay:** 网络传输过程当中延迟的时间，单位 10^{-6} 秒
- **offset:** 时间补偿的结果，单位 10^{-3} 秒
- **jitter:** Linux 系统时间与 BIOS 硬件时间的差异时间，单位 10^{-6} 秒

4、要让 NTP Server/Client 正常工作，要注意以下几点

- 上述的 `ntpstat` 以及 `ntpq -p` 的输出结果中，NTP 服务器确实要能够连接上层 NTP 才行，否则客户端将无法对 NTP 服务器进行同步更新
- NTP 服务器时间不可与上层差异太多，否则上层服务器不会将正确的时间传过来
- 特别注意服务器防火墙在 UDP port 123 有没有开
- 等待的时长

15.2.4. 安全性设置

1、`/etc/ntp.conf` 中设置 NTP 这个 daemon 的服务限制范围

2、还需要修订防火墙规则

```
iptables -A INPUT -i $EXTIF -p udp -s 192.168.200.0/24 --dport 123 -j ACCEPT
```

15.3. 客户端的时间更新方式

- 1、如果仅有不到 10 台主机，其实没有假设 NTP 服务器的需求，只要能够在主机上面以 NTP 客户端软件来进行网络校时就可以同步化时间了
- 2、如果是群集计算机或登录服务器(NIS)，那使用时间服务器就比较好

15.3.1. Linux 手工校时工作：`date`、`hwclock`

1、软件时钟与硬件时钟

- 软件时钟：Linux 自己的系统时间，从 1970 年 1 月 1 日开始记录的时间参数
- 硬件时钟：计算机系统在 BIOS 记录的实际时间，这也是硬件所记录的
- 软件时钟可以通过 `date` 这个命令来修改，硬件时钟需要通过 `hwclock` 这个命令来写入

2、`<date>`

- `date MMDDhhmmYYYY`
- **MM:** 月份
- **DD:** 日期
- **hh:** 小时
- **mm:** 分钟
- **YYYY:** 公元年

3、`<hwclock>`

- **-r:** 也就是 `read`，读出目前 BIOS 内的时间参数
- **-w:** 也就是 `write`，将目前的 Linux 系统时间写入 BIOS 内

15.3.2. Linux 的网络校时

1、在 Linux 环境中利用 NTP 客户端程序，即 `ntpdate` 这个程序就能够进行时间的同步化

➤ NTP 服务器本来就会与上层时间服务器进行时间的同步化，**所以在默认情况下，NTP 服务器不可以使用 ntpdate**

➤ 也就是说 `ntpdate` 与 `ntpd` 不能同时启用

2、`<ntpdate>`

➤ `ntpdate [-dv] [NTP IP/hostname]`

➤ `-d`: 进入排错模式(debug)，可以显示出更多的有效信息

➤ `-v`: 显示更详细的信息

Chapter 16. 文件服务器之二：SAMBA 服务器

1、如果想要共享文件

- 在 Linux 对 Linux 的环境下，最简单的方法就是通过 NIS
- 在 Windows 对 Windows 的环境下，最简单的方法则是"网上邻居"
- 如果局域网中既有 Windows 也有 Linux 而且想要共享文件系统的话，可以使用 SAMBA 服务器，SAMBA 可以让 Linux 加入 Windows 的网上邻居支持

16.1. 什么是 SAMBA

16.1.1. SAMBA 的发展历史与名称的由来

1、早期，文件数据在不同主机之间的传输大多是使用 FTP 这个服务器软件

- FTP 有个小问题：无法直接修改主机上面的文件数据
- 如果想要修改 Linux 主机上面的某个文件时，必须要从服务器下载，此时服务器与客户端都存在，万一忘记上传了，过段时间就不知道哪个文件的版本更新了

2、让文件在两部主机之间直接修改：NFS 与 CIFS

- NFS：在客户端将 Server 所提供的共享目录挂载进来，那么在客户端的机器上面可以直接使用 Server 上的文件资料了
- CIFS(Common Internet File System)：CIFS 最简单的用途就是网上邻居
 - Windows 系统的计算机可以通过桌面上"网上邻居"来共享别人所提供的文件数据

3、利用数据包检测逆向工程发展的 SMB Server

- 通过 DOS 与 DEC 的 Unix 系统在进行数据共享时所使用的通信协议信息，逆向开发出 Server Message Block(SMB)这个文件系统

4、强调

- 在 Unix Like 上面可以共享文件资料的 file system 是 NFS
- 在 Windows 上面使用的"网上邻居"所使用的文件系统则称为 Common Internet File System(CIFS)

5、取名为 SAMBA 的主因

- SMB 是没有意义的字，无法注册
- SAMBA 刚好包含 SMB，又是拉丁舞蹈的名称，因此可以拿来注册

16.1.2. SAMBA 常见的应用

1、SAMBA 的主要功能

- 共享文件与打印机服务
- 可以提供用户登录 SAMBA 主机时的身份认证，以提供不同身份用户的个别数据
- 可以进行 Windows 网络上的主机名解析(NetBIOS Name)
- 可以进行设备的共享

2、利用软件直接编辑 WWW 主机上面的网页数据

- 如果安装了 SAMBA 服务器的话，通过"网上邻居"功能，直接远程连接服务器所提供的目录，就可以直接在个人计算机上面修改主机的文件数据，并且只有一份正确的数据，类似在线编辑

3、做成可直接连接的文件服务器

- 用 SAMBA 将硬盘空间共享出来，由于用于在登录 SAMBA 这个服务器主机时需要输入用户数据(账号与密码)，不同的登陆者会取得不一样的目录资源，因此可以避免自己的数据在公用计算机上面被窥视

4、打印机服务器

- Linux 作为服务器比较稳定
- 常见的攻击手法均是针对 Windows 而来的

16.1.3. SAMBA 使用的 NetBIOS 通信协议

1、就像 NFS 是架构在 RPC Server 上面一样，SAMBA 这个文件系统是架构在 NetBIOS 这个通信协议上面所开发出来的

2、最早 IBM 发展 NetBIOS 的目的仅是要让局域网内少数计算机进行网络连接的通信协议而已，考虑的角度并不是大型网络，因此 NetBIOS 是无法跨路由的

- SAMBA 最早发展的时候，是想要 Linux 系统可以加入 Windows 的系统当中来共享使用彼此的文件数据，因此 SAMBA 就架构在 NetBIOS 发展出来了
- 由于 NetBIOS 是无法跨路由的，因此使用 NetBIOS 发展起来的服务器理论上也是无法跨越路由的
- 可以通过 NetBIOS over TCP/IP 技术(将 NetBIOS 类比成明信片，TCP/IP 类比成邮件系统)，可以跨路由使用 SAMBA 服务器所提供的功能

16.1.4. SAMBA 使用的 daemons

1、由于 NetBIOS 发展初期就着眼在局域网内快速数据交流，因此没有使用类似 TCP/IP 的传输协议，也就不需要 IP 的设置

- 主机在 NetBIOS 协议当中定义为使用"NetBIOS Name"，每一台主机必须要有不同的 NetBIOS Name 才行，而文件数据就是在不同的 NetBIOS Name 之间沟通

2、取得对方主机的 NetBIOS Name 定位该主机所在

- 想要登录某个 Windows 主机使用它所提供的文件数据时，必须要加入该 Windows 主机的工作组(Workgroup)，并且我们的机器也必须要设置一个主机名，这个主机名与 Hostname 是不同的
- 因为这个主机名是架构在 NetBIOS 协议上的，我们可以简单称其为 NetBIOS Name，**在同一个组中，NetBIOS Name 必须独一无二**

3、利用对方给予权限访问可用资源

- 当我们找到该主机名后，是否能登录该对方主机或者是使用对方主机所提供的资源，还要看对方 Windows 主机有没有提供我们使用的权限
- **SAMBA 通过两个服务来控制这个步骤**
 - nmbd：这个 daemon 是用来管理工作组、NetBIOS Name 等的解析，主要利用 UDP 协议开启 port 137、138 来负责名称解析的任务
 - smbd：这个 daemon 的主要功能就是用来管理 SAMBA 主机共享的目录，文件与打印机，主要利用可靠的 TCP 协议来传输数据，开放的端口为 139 以及 455

16.1.5. 连接模式的介绍 (Peer/Peer、Domain model)

1、Peer/Peer(Workgroup model，对等模式)

- 在局域网里面所有 PC 均可以在自己的计算机上面管理自己的账号与密码，同时每一台计算机也都具有独立执行各项软件的能力，只是通过网络将各个 PC 连接在一起的架构，每一台机器都是可以独立运行的

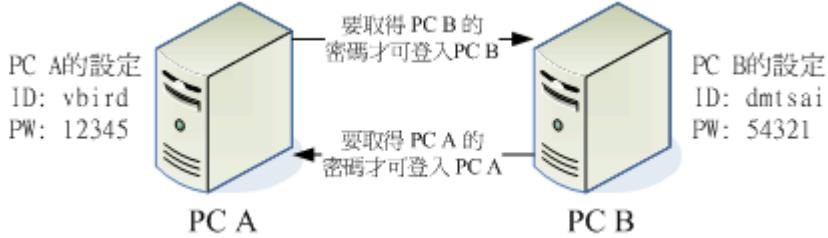


图 16-1 Peer/Peer 连接的示意图

- Peer/Peer 的架构的好处是每台计算机均可以独立运作，而不受他人影响
- Peer/Peer 的架构的缺点就是当整个网络内所有人员都要进行数据共享时，光知道所有计算机里面的账号与密码，就会很伤脑筋
- 因此 Peer/Peer 架构比较适合小型的网络，或者是不需要常常进行文件数据共享的网络环境，或者是每个用户都独自拥有该计算机的拥有权(也就是说，该计算机是用户的，而不是公用的)

2、Domain model(主控模式)

- 将所有的账号与密码都放置在一台主控计算机(Primary Domain Controller, PDC)上面
- 任何想要使用任何计算机时，都需要在屏幕前方输入账号与密码，然后全部利用 PDC 服务器的辨识之后，才给予适当的权限，**也就是说，不同的身份还具有不一样的计算机资源**

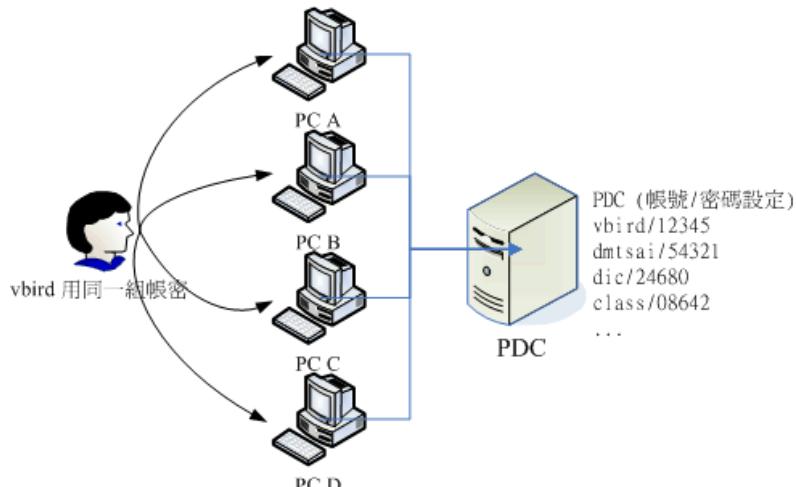


图 16-2 Domain model 连接的示意图

16.2. SAMBA 服务器的基础设置

16.2.1. SAMBA 所需软件及其软件结构

1、需要的软件

- samba: 这个软件主要提供了 SMB 服务器所需要的各项服务程序(smbd 以及 nmbd)，相关的文件以及其他与 SAMBA 相关的 logrotate 配置文件及开机默认选项文件等
- samba-client: 这个软件提供了当 Linux 作为 SAMBA Client 端时，所需要的

工具命令，例如挂载 SAMBA 文件格式的 mount.cifs，取得类似网上邻居相关树形图的 smbtree 等

- samba-common: 这个软件提供的则是服务器与客户端都会使用到的数据，包括 SAMBA 主要配置文件(smb.conf)、语法检验命令(testparm)等
- 利用 yum 进行安装即可

2、与 SAMBA 的软件架构相关的配置文件

- /etc/samba/smb.conf
- /etc/samba/lmhosts
- /etc/sysconfig/samba
- /etc/samba/smbusers
- /var/lib/samba/private{passdb.tdb,secrets.tdb}
- /usr/share/doc/samba-<版本>

3、常用的脚本方面，若分为服务器和客户端功能，则主要有下面几个数据

- /usr/sbin/{smdb,nmbd}: **服务器功能**，就是最重要的权限管理(smdb)以及 NetBIOS Name 查询(nmbd)两个重要的服务程序
- /usr/bin/{tdbdump,tdbtool}: **服务器功能**
 - 在 SAMBA 3.0 以后的版本中，用户的账号与密码参数已经转为使用数据库了
 - SAMBA 使用的数据库名称为 TDB(Trivial DataBase)
 - 既然使用数据库，就要用数据库的控制命令来处理
 - tdbdump 可以查看数据库的内容，tdbtool 则可以进入数据库操作接口直接手动修改账户及密码参数，需要安装 tdb-tools 这个软件
- /usr/bin/smbstatus: **服务器功能**，可以列出当前 SAMBA 的连接状况，包括每一条 SAMBA 连接的 PID，共享的资源，使用的用户来源等
- /usr/bin/{smbpasswd,pdbedit}: **服务器功能**，在管理 SAMBA 的用户账号密码时，早期使用 smbpasswd，后来由于改用 TDB 数据库，因此使用 pdbedit 命令来管理用户数据
- /usr/bin/testparm: **服务器功能**，主要在检查配置文件 smb.conf 的语法正确与否，**当你编辑过 smb.conf 时，务必使用这个命令来检查一次**，避免因为打字错误引起困扰
- /sbin/mount.cifs: **客户端功能**
 - 在 Windows 上，我们可以设置"网络驱动"来连接到自己的主机
 - 在 Linux 上面，我们通过 mount(mount.cifs)来将远程主机共享的文件与目录挂载到自己的 Linux 主机上面
- /usr/bin/smbclient: **客户端功能**，当 Linux 主机想要通过"网上邻居"的功能来查看其它计算机所共享出来的目录与设备时，就可以使用 smbclient 查看，也可以使用在自己的 SAMBA 主机上面，查看是否设置成功
- /usr/bin/nmblookup: **客户端功能**，类似 nslookup，重点在查出 NetBIOS Name
- /usr/bin/smbtree: **客户端功能**，这个配置文件有点像 Windows 系统的"网上邻居"显示的结果，可以显示类似"靠近我的计算机"之类的数据，能够查到与工作组与计算机名称的树形目录分布图

16.2.2. 基础的网上邻居共享流程与 smb.conf 的常用设置项目

1、网上邻居的默认限制如下

- 服务器与客户端之间必须要在同一个网络中(否则需要修改 Windows 默认防火墙)
- 最好设置为同一工作组
- 主机名称不可相同(NetBIOS Name)
- 专业版 Windows XP 最多仅能提供 10 个用户连接到同一台网上邻居服务器上

2、SAMBA 怎么设置

- 1) 服务器全局设置方面: 在 `smb.conf` 中设置好工作组, NetBIOS 主机名、密码使用状态(无密码共享货本机密码)等
- 2) 规划准备共享的目录参数: 在 `smb.conf` 内设置好预计要共享的目录或设备以及可供使用的账号数据
- 3) 建立所需要的文件系统: 根据步骤 2 的设置, 在 Linux 文件系统中建立好共享出去的文件或设备, 以及相关的权限参数
- 4) 建立可用 SAMBA 的账号: 根据步骤 2 的设置, 建立所需的 Linux 实体账号, 再以 `pdbeedit` 建立使用 SAMBA 的密码
- 5) 启动服务: 启动 SAMBA 的 `smbd` 服务, 开始运转

3、`smb.conf` 的配置

- 这个文件可以分为两个部分: **主机信息部分与共享信息**
- 语法注意点
 - 在 `smb.conf` 当中, ;井字号与分好(#与;)都是注释符号
 - 在这个配置文件中, 大小写是没关系的, 以你为 Windows 不分大小写
- `smb.conf` 的服务器全局参数: `[global]` 项目
 - 全局参数包括: 工作组、主机的 NetBIOS 名称、字符编码的显示、日志文件的设置、是否使用密码以及使用密码验证的机制等
 - 关于主机名信息方面的参数
 - `workgroup=`工作组名称 <==工作组要相同
 - `netbios name=`主机的 NetBIOS 名称 <==每部主机不同
 - `server string=`主机的简易说明 <==随便写
 - 语言转换设置
 - `display charset=`自己服务器上面的显示编码 <==一般来说与 unix charset 相同
 - `unix charset=`在 Linux 服务器上面使用的编码, 参考`/etc/locale.conf` 的设置, CentOS7 之前的配置文件是`/etc/sysconfig/i18n`
 - `dos charset=`Windows 客户端的编码, 一般来说简体中文使用的是 GB 2312 编码, 这个编码在 SAMBA 内的格式称为"cp936"
 - 参考 <http://phorum.vbird.org/viewtopic.php?t=22001>
 - 登录方面的设置
 - `log file=`日志存储的文件 <==文件名可能会使用变量处理
 - `max log size=`日志文件最大能达到多少 Kbytes, 若大于该数字, 则会被 `rotate` 掉
 - 密码参数设置
 - `security=share/user/domain` <==三选一, 这三个设置值分别代表
 - ◆ `share`: 共享的数据不需要密码, 大家均可使用(没有安全性)
 - ◆ 使用 SAMBA 服务器本身的密码数据库, 密码数据库与 `passdb`

- backend 有关
 - ◆ domian: 使用外部服务器的密码, 也就是 SAMBA 是客户端之意, 需要提供 password server=IP 的设置值才行
 - encrypt passwords=Yes <==代表需要加密
 - passdb backend=数据库格式 <==密码文件已经转为使用数据库了, 默认的数据格式为 tdb, 默认的文件则放置到 /var/lib/samba/private/passwd.tdb
- smb.conf 的服务器的共享资源的相关参数设置[共享的名称]
 - [共享名称]: "代号"而已
 - comment: 这个目录的说明而已
 - path: 这个共享名称实际会进入 Linux 文件系统, 在网上邻居中看到的是[共享]的名称, 而实际操作的文件系统则是在 path 里面所设置的
 - browsable: 是否让所有的用户都看到这个项目
 - writable: 是否可以写入?
 - **如果 writable 设置为 yes, 同时 read only 也是 yes, 那么后出现的设置值为主要设置值**
 - create mode 与 directory mode: 与权限有关
 - writelist=用户, @组: 这个项目可以指定能够进入到此资源的特定用户, 如果是@group 的格式, 则加入改组的用户均可取得使用的权限, 设置上比较简单
- smb.conf 内的可用变量功能
 - %S: 取代当前的设置项目值, 所谓"设置项目值"就是在[共享]里面的內容

[homes]

 - valid users=%S
 - 因为 valid users 是允许的登陆者, 设置为%S 表示任何可登陆的用户都能够登陆的意思, 如果 dmtsaI 这个用户登录之后, 那么[homes]就会变成[dmtasi]
 - %S 的用意就是替换掉当前[]里面的内容
 - %m: 代表 Client 端的 NetBIOS 主机名
 - %M: 代表 Client 端的 Internet 主机名, 就是 hostname
 - %L: 代表 SAMBA 主机的 NetBIOS 主机名
 - %H: 代表用户的主目录
 - %U: 代表当前登录的用户的用户名
 - %g: 代表登录的用户的组名
 - %h: 代表这部 SAMBA 主机的 Hostname, 注意是 Hostname 不是 NetBIOS Name
 - %l: 代表 Client 的 IP
 - %T: 代表当前的日期与时间

16.2.3. 不需要密码的共享 (security=share, 纯测试)

- 1、可以不需要密码就能够使用 SAMBA 主机所提供的目录资源
- 2、假设条件

- LAN 内所有网上邻居主机的工作组(workgroup)为 liuyehouse

- 这台 SAMBA 服务器的 NetBIOS 名称(netbios name)为 liuyeserver
- 用户认证等级设置(security)为 share
- 取消原本有共享的[homes]目录
- 仅共享/tmp 这个目录，并且取名为 temp
- Linux 服务器的编码格式假设为国际通用码(Unicode，亦即 utf8)
- 客户端为中文 Windows，在客户端的软件也使用 GB2312 编码

3、设置 smb.conf 配置文件

- 先检查 Linux 的语言是否为 utf-8，后缀有 utf-8 的都是(zh_CN.UTF-8)，配置文件是/etc/locale.conf
- cd /etc/samba
- cp smb.conf smb.conf.raw <==先备份，避免改坏了
- vim smb.conf

4、用<testparm>检查 smb.conf 配置文件语法设置的正确性

- testparm -v
- -v：查阅完整的参数设置，连同默认值也会显示出来

5、服务器端的服务启动与端口观察

- systemctl restart smb.service
- netstat -tunlp | grep mbd
- SAMBA 当中默认会启动多个端口，包括 TCP 端口(139,145)，以及进行 NetBIOS 名称解析之类工作的 UDP 端口 137,138

6、假设本机为客户端的测试(默认用 lo 接口)

- <smbclient> -L ///[IP|hostname] [-U 用户账号]
- -L：仅查阅后面接的主机所提供共享的目录资源
- -U：以后面这个账号来尝试访问该主机可使用的资源
- smbclient -L //127.0.0.1
- mount -t cifs //127.0.0.1/temp /mnt
- <未完成>：/etc/samba/smb.conf 中 security 无法设置为 share 的值

16.2.4. 需要账号密码才可登陆的共享(security=user)

1、SAMBA 用户账号必须要存放于 Linux 系统当中(/etc/passwd)，但是 SAMBA 的密码与 Unix 的密码文件并不相同(这是因为 Linux 与网上邻居的密码验证方式以及编码风格不同导致的)

2、假设条件

- 用户认证等级设置(security)为： user
- 用户密码文件使用 TDB 数据库格式，默认文件在/var/lib/samba/private/内
- 密码必须加密
- 每个可使用 SAMBA 的用户均拥有自己的用户主目录
- 设置三个用户，名称为 smb1、smb2、smb3，且君附属于 users 用户组，这三个用户的 Linux 密码为 1234，SAMBA 密码为 4321
- 共享/home/project 这个目录，且共享资源为 project
- 加入 users 这个组的用户可以使用//IP/project，且在该目录下 users 这个组的用户具有写入的权限

3、smb.conf 配置文件与目录权限的相关配置

[global]

```

workgroup = liyehouse
netbios name=liyeserver
server string = THis is liuye's samba server %v
unix charset=utf8
dos charset=cp36
log file = /var/log/samba/log.%m
max log size = 50
load printers = no

security = user
passdb backend=tdbsam

[homes]
comment =Home Directories
browseable=no
writable=no
create mode=0664
directory mode=0775

[project]
comment=sumuser's project
path=/home/project
browseable=yes
writable=yes
write list=@users

```

➤ 别忘了创建共享目录

- mkdir /home/project
- chgrp users /home/project
- chmod 2770 /home/project

4、设置可使用 SAMBA 的用户账号与密码

➤ Linux 的文件系统与 SAMBA 设置的用户登录权限的相关性

- 在 Linux 系统下，任何程序都需要取得 UID 和 GID(User ID 和 Group ID)的身份后，才能够拥有该身份的权限，也才能够适当地进行访问文件等操作
- 关于 Linux 这个系统的 UID、GID 与账号的对应关系，一般记录在 /etc/passwd 当中，也可以通过 NIS、LDAP 等方式来进行对应
- **SAMBA 仅是 Linux 下的一套软件，使用 SAMBA 来访问 Linux 文件系统时，还是需要以 Linux 系统下的 UID 和 GID 为准则**

➤ 我们需要通过 SAMBA 提供的功能来进行 Linux 的访问，而 Linux 的访问时需要取得 Linux 系统上面的 UID 与 GID 的，因此，我们登录 SAMBA 服务器时，所利用 SAMBA 取得的其实是 Linux 系统里面的相关账号

- **也就是说，在 SAMBA 上面的用户账号，必须要是 Linux 账号中的一个**
- 因此，不考虑 NIS 或 LDAP 等其他账号验证方式，SAMBA 服务器所提供的可登陆的账号名称，必须要存在于/etc/passwd 中

➤ 创建 Linux 账号

- useradd -G users umb1

- useradd -G users umb2
 - useradd -G users umb3
 - echo 1234 | passwd --stdin smb1
 - echo 1234 | passwd --stdin smb2
 - echo 1234 | passwd --stdin smb3
- <pdbeedit>
- pdbeedit -L [-vw] <==单纯查看账户信息
 - pdbeedit -a | -r | -x -u 账号 <==添加/修改/删除账号
 - pdbeedit -a -m -u 机器账号 <==与 PDC 有关的机器码
 - -L: 列出目前在数据库当中的账号与 UID 等相关信息
 - -v: 需要搭配-L 来执行, 可列出更多的信息, 包括用户主目录等数据
 - -w: 需要搭配-L 来执行, 使用旧版的 smbpasswd 格式来显示数据
 - -a: 添加一个可使用的 SAMBA 账号, 后面的账号需要在/etc/passwd 内存在
 - -r: 修改一个账号的相关信息, 需搭配很多特殊参数
 - -x: 删除一个可使用的 SAMBA 账号, 可先用-L 找到账号后再删除
 - -m: 后面接的是机器的代码, 与 domain model 有关
- 开始添加用户
- pdbeedit -a -u smb1 <==需要输入该账号作为 SAMBA 账号的密码
 - pdbeedit -a -u smb2
 - pdbeedit -a -u smb3
- 管理 TDB 数据库格式建议使用 pdbeedit 来处理, smbpasswd 用于修改密码

5、重新启动 SAMBA 并进行自我测试

- systemctl restart smb.service
- smbclient -L //127.0.0.1 <==匿名以查阅的方式登录
- smbclient -L //127.0.0.1 -U smb1 <==用 smb1 以查阅的方式登录
- mount -t cifs //127.0.0.1/smb1 /mnt **-o username=smb1**
- ls -a /mnt
- ls: reading directory /mnt: Permission denied
 - 这是由于 SAMBA 是会对外提供服务的, 因此 SELinux 会特别关照一下这个服务, 包括默认用户主目录不会有开放的权限, 默认的 SELinux type 不对就无法使用
 - **setsebool -P samba_enable_home_dirs=1**

6、关于权限的再说明与累加其他共享资源的方式

- 有时候, 明明在 smb.conf 中设置了 writable 可写入, 用户登录身份也没有问题, 但是就是无法挂载或写入, 主要问题常常来自于 Linux 文件系统的权限
- **如果还想要扩充共享的目录与能够登录的用户, 依照以下步骤进行:**
- 利用编辑 smb.conf 来开放其他的目录资源, 并且特别注意 Linux 在该目录下的权限, 请使用 chown 与 chmod
 - 利用 pdbeedit 来添加其他可用 SAMBA 账号, 如果该账号并没有出现在 /etc/passwd 里面, 请先 useradd 添加账号
 - 不论进行完任何设置, 请先 testparm 进行确认, 然后 systemctl restart smb.service 重启服务

7、Window 客户端的连接

- 开始->运行
- 输入\\192.168.200.254\homes
- 输入账号密码，例如 smb1, 4321
- 记得设置防火墙以及设置 SELinux 为宽容模式(setenforce permissive)

16.2.5. 设置成为打印机服务器 (CUPS 系统)

1、现在打印机的网络功能已经很强了，每台打印机可以独立作为各个 PC 的独自

的打印机，也没有必要进行 SAMBA 的网络打印机服务

2、我又没有打印机，玩个卵???

16.2.6. 安全性的议题与管理

1、使用 SAMBA 其实是由一定程度的危险性的，因为很多网络攻击的蠕虫，病毒，木马就是通过网上邻居来攻击的

2、为了阻挡不必要的连接，所以 CentOS 的 SELinux 已经关闭了很多 SAMBA 连接的功能，因此默认情况下，很多客户端的挂载可能会有问题

3、此外，仅开放有权限的网络来源，以及通过 smb.conf 来管理特定的权限，也是很重要的

16.2.7. SELinux 的相关议题

1、找出与 SAMBA 有关的 SELinux 规则

- getsebool -a | grep samba
 - samba_create_home_dirs --> off
 - samba_domain_controller --> off <==PDC 时可能会用到
 - samba_enable_home_dirs --> off <==开放用户使用用户主目录
 - samba_export_all_ro --> off <==允许只读文件系统的功能
 - samba_export_all_rw --> off <==允许读写文件系统的功能
 - samba_load_libgfapi --> off
 - samba_portmapper --> off
 - samba_run_unconfined --> off
 - samba_share_fusefs --> off
 - samba_share_nfs --> off
 - sanlock_use_samba --> off
 - tmpreaper_use_samba --> off
 - use_samba_home_dirs --> off <==类似用户的用户主目录的开放
 - virt_sandbox_use_samba --> off
 - virt_use_samba --> off
- 几乎所有的规则默认都是关闭的
- setsebool -P samba_enable_home_dirs=1 <==这样就可以访问主目录了
- 另外，共享成为 SAMBA 的目录还需要有 samba_share_t 的类型
 - chcon -t samba_share_t /home/project
- 如果共享的目录不只是 SAMBA，还包括 FTP 或者其他的服務，那就可能需要使用 public_content_t 这个大家都能读取的类型才行
- 如果还发现有问题，依照/var/log/messages 里面的信息去修改

2、利用 iptables 来管理

- 最简单的管理登录 SAMBA 的方法就是通过 iptables 了
- 需要加上以下几条放行规则
 - `iptables -A INPUT -i $EXTIF -p tcp -s [Ip 或网段] -m multiport \> --dport 139,455 -j ACCEPT`
 - `iptables -A INPUT -i $EXTIF -p udp-s [Ip 或网段] -m multiport \> --dport 137,138 -j ACCEPT`
- 由于 `smbd` 以及 `nmbd` 并不支持 TCP Wrappers，所以只能通过 `iptables` 来控制

3、防火墙议题：通过内建的 SAMBA 设置(smb.conf)

- SAMBA 已经有许多防火墙机制，就是在 `smb.conf` 内的 `hosts allow` 及 `hosts deny` 这两个参数
- 通常我们只需要 `hosts allow` 即可，没有写入这个设置项目的其他来源就会被拒绝，这一点与其他{allow,deny}不同
- `vim /etc/samba/smb.conf`
`hosts allow=127. 192.168.100.254 192.168.1`
 - 这个设置值的内容支持部分比对，网段只需要写出表示网段的数值即可
- 在防火墙方面，只需要使用 `iptables` 或者 `hosts allow` 其中一项即可，推荐使用 `hosts allow`

4、文件系统议题：利用 Quota 限制用户磁盘使用

16.2.8. 主机安装时的规划与中文扇区挂载

1、由于`/home` 被共享，因此`/home` 可能会不够用，可以使用大一点的磁盘，或者使用磁盘阵列，使用 LVM

2、规划

- 在安装 Linux 的时候，建议不安装 X Windows
- 在规划 Linux 时，`/home` 最好独立出一个 partition，而且磁盘空间最好大一些
- `/home` 独立出来的 partition 可以单独进行 quota 的作业，以规范用户的最大硬盘用量
- 无网卡的打印机(USB)可以直接连接到 Linux 主机再通过 SAMBA 共享
- 由于 SAMBA 一般都仅针对内部(LAN)主机进行开放，所以 SAMBA 直接使用 private IP 来设置即可

3、另外，如果 SAMBA 服务器需要挂载含有中文的 partition 时，譬如说将原本的 Windows XP 的 FAT32 文件系统挪到 Linux 系统下，此时如果用一般模式来挂载该分区，一些中文文件名可能会无法被顺利地显示出来

- `mount -t vfat -o iocharset=utf8,codepage=936 /dev/sd[a-p][1-15] /mount/point`

16.3. SAMBA 客户端软件功能

1、假设局域网内有 Windows/linux 系统，这两种系统都通过 NetBIOS over TCP/IP 来连上 SAMBA 服务器的

- 在局域网内的主机最好具有相同的工作组，具有不同的主机名

- 可以在网上邻居当中看到的通常是相同组的主机
- Windows 的网上邻居默认仅有同一网段的主机才能登陆(Windows 防火墙设置)

16.3.1. Windows 系统的使用

- 1、在"网络"找不到有共享的 Linux 主机
- 2、让 Windows 系统上的网上邻居支持不同网络的 IP 连接
 - 控制面板-->系统和安全-->Windows 防火墙-->高级设置
 - 入站规则，中间列表找到"文件和打印机共享(SMB-In)"，右边方框内点属性-->作用域-->添加
- 3、通过 port445 的特殊登录方式
 - 若 SAMBA 服务器启动 port445，且它已经共享了某个目录，例如 project 这个**共享资源名**
 - 那么这个目录的完整写法为：\\192.168.200.254\project
 - 通过开始-->运行"\192.168.200.254\project"即可访问
 - 删除会话：net use * /delete

16.3.2. Linux 系统的使用

- 1、smbclient：查询网上邻居共享的资源，以及使用类似 FTP 的方式上传/下载网上邻居
 - 主要通过 smbclient 来观察，再以 mount 来挂载文件系统
 - <smbclient>
 - smbclient -L // [IP | hostname] [-U username]
 - smbclient ' // [IP | hostname] / 资源名' [-U username] <单引号不要也行
 - 直接以类似 FTP 的方式登录远程主机
 - 可以使用 dir、get、put 等常用的 FTP 命令来进行数据传输
- 2、mount.cifs：直接挂载网上邻居称为网络驱动器
 - 事实上，smbclient 不太方便，因为使用的是 ftp 的功能语法，较奇怪
 - 早期的 SAMBA 主要提供 smbmount 或 mount.smbfs 这个命令来挂载 (smbfs 是 SMB filesystem 的缩写)，不过这个命令已经被可以进行比较好编码判断的 mount.cifs 所取代
 - mount.cifs 可以将远程服务器共享出来的目录整个挂载到本机的挂载点，如此一来，远程服务器的目录就好像在本机上的分区一样，可以直接进行复制、编辑等动作
 - mount -t cifs //IP/共享资源 /挂载点 [-o options]
 - -o 后面接的参数通常为
 - username=登录账号
 - password=登录密码，没有该项的话，执行命令后需要输入密码
 - iocharset=本机的语言编码方式
 - codepage=远程主机的语言编码方式
- 3、nmblookup：查询 NetBIOS Name 与 IP 及其他相关信息
 - nmblookup [-S] [-U wins IP] [-A IP] name
 - -S：除了查询 name 的 IP 之外，亦会找出该主机的共享资源与 MAC 等
 - -U：后面一般接 Windows 的主要名称管理服务器的 IP，可与-R 互用

- -R: 与-U 互用, 以 Wins 服务器来查询某个 NetBIOS Name
- -A: 相对于其他参数, -A 后面可接 IP, 通过 IP 找出相对的 NetBIOS 数据
- <未完成>???

4、<smbtree>: 网上邻居浏览器显示方式

- 想要使用类似 Windows 上面, 可以一看就明了各个网上邻居所共享的资源时, 可以使用 smbtree 来直接查询
- smbtree [-bDS]
- -b: 以广播方式取代主要浏览器的查询
- -D: 仅列出工作组, 不包括共享的资源
- -S: 列出工作组与该工作组下的计算机名称, 不包括各项资源目录

5、<smbstatus>

- 观察 SAMBA 的状态, 算是服务器的相关功能, 主要目的是查看目前 SAMBA 有多少人来连接

16.4. 以 PDC 服务器提供账号管理

1、PDC 可以让用户在计算机教室的任何一个地方, 都用同一组账号密码登录, 并可以取得相同的用户主目录等数据, 这与 NIS 搭配 NFS 是很类似的

16.4.1. 让 SAMBA 管理网络用户的一个实例案例

1、例如有一个计算机教室里面有 50 台 Windows XP Pro 个人计算机

- 由于计算机教室大家都会使用, 因此这 50 台计算机有使用还原精灵, 也就是每次计算机重新启动后整个操作系统就会还原成原本的样子
- 但我们希望我们有用户主目录, 不然重启后就丢失了
- 因此我们可以用一台主机来让他们存储数据, 那便是 PDC 服务器(Primary Domain Controller)

2、SAMBA PDC 的作用很简单, 就是让 SAMBA PDC 成为整个局域网的域管理员 (domain controller), 然后让 Windows 主机加入这个域

- 未来用户利用 Windows 登录时, Windows 会前往 PDC 服务器取得用户的账号密码, 同时 PDC 还会传送用户的重要数据到个人的 Windows 计算机上
- Windows 计算机注销后, 该用户修改过的数据也会返回给 PDC, 如此一来不管这个用户在哪一台个人计算机上登录, 它都能取得正确的个人资料

<未完成>: 没有空间装 Windows 虚拟机了, 下次再说吧

Chapter 17. 局域网控制者：Proxy 服务器

17.1. 什么是代理服务器

1、代理服务器(Proxy)的原理其实很简单，就是一类似代理人的身份去取得用户所需要的数据

- 但是由于它的"代理"能力，使得我们可以通过代理服务器来实现防火墙功能与用户浏览数据的分析
- 此外，也可以利用代理服务器来实现节省带宽的目的，以及加快内部网络对因特网 WWW 的访问速度
- 代理服务器对于企业来说，是一个很不错的选择

17.1.1. 什么是代理服务器

1、代理服务器最主要的功能：当客户端有因特网的数据要求时，Proxy 会帮用户去向目的地取得用户所需要的数据

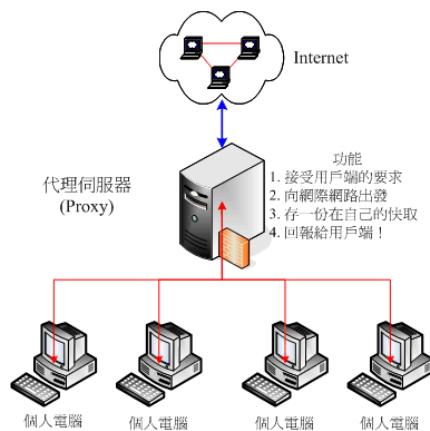


图 17-1 代理服务器、客户端与因特网的关系示意图

2、一般来说，代理服务器会搭建在整个局域网的单点对外防火墙上，而在局域网内部的计算机就都是通过 Proxy 来向因特网要求数据

3、在 Proxy 与客户端的关系当中，客户端向外部要求的数据事实上都是 Proxy 帮用户取得的

- 因此因特网上面看到的要求数据者，将会是 Proxy 服务器的 IP 而不是客户端的 IP
- 4、除此之外，Proxy 还有一个很棒的功能，那就是防火墙的功能
 - 由于客户端的个人计算机要连上因特网一定要经过 Proxy 服务器
 - 如果有人想要入侵客户端系统时，由于 Proxy 在最外部，所以攻击者就会攻击错方向，如此一来就比较安全了

17.1.2. 代理服务器的工作流程

1、当客户端指定了代理服务器之后，客户端想要取得因特网上面的信息时工作流程如下

➤ 当 Proxy 的缓存拥有用户所想要的数据包时(Step1-4)

- 1) Client 端向 Server 端发送一个数据需求数据包
- 2) Server 端接收之后，先比对这个数据包的来源与预计要前往的目标网站

是否为可接受?如果来源与目标都是合法的，或者说，来源与目标网站 Proxy 都能帮忙取得数据时，那么 Server 端会开始替 Client 取得数据，这个步骤比较重要的就是"比对政策"，有点像认证

- 3) Server 首先会检查自己缓存(新的数据可能在内存中，较旧的数据则放置在硬盘上)数据，如果有 Client 所需的数据，那就将数据取出，而不经过向 Internet 要求数据的过程

- 4) 最后将数据发送给 Client 端

➤ **当 Proxy 的缓存没有用户所想要的数据包时(Step1-5)**

- 1) Client 端向 Server 端发送一个需求数据包
- 2) Server 端接受之后，开始进行数据比对
- 3) Server 发现缓存并没有 Client 所需要的数据时，准备前往因特网获取数据
- 4) Server 开始向 Internet 发送要求与取得相关数据
- 5) 最后将数据回送给 Client 端

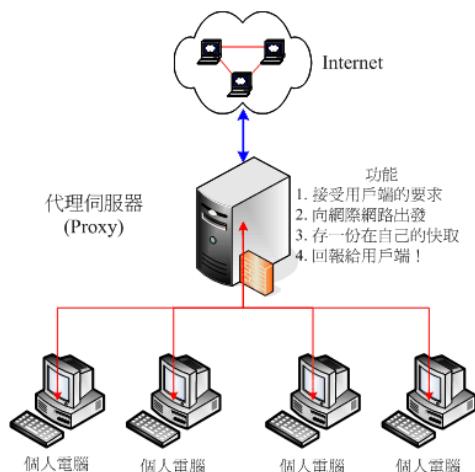


图 17-2 代理服务器的工作流程图：缓存数据与客户端

- 当 Proxy 曾经帮助某用户取得 A 数据后，当后来的用户想要重复取得 A 数据时，那么 Proxy 就会从自己的缓存里将 A 数据取出传送给用户，而不用到因特网去取得同样的这份资料
- 由于忽略了步骤 4，感觉上网络速度变快了，但其实只是直接从 Proxy 的缓存里取得而已
- 在目前的网络社会里，由于宽带技术已经很成熟，所以在不乱用的情况下，网络带宽理论上是足够的，那么使用 Proxy 之后效率**并不会提升**
- 因为 Proxy 会常常去读取硬盘内的数据
 - 而硬盘内的缓存数据又是通过某种特殊方式管理，因此要找到该份数据就要花一些事件
 - 如果硬件效率不佳时，那么 Proxy 反而可能会让你感觉网络传输不流畅

17.1.3. 上层代理服务器

- 1、Proxy 还可以指定另外一台 Proxy'当成本台 Proxy 的代理服务器，例如下图

- Local Proxy 并不会主动去 Internet 获取数据，而是再通过上层代理服务器向 Internet 要求数据
- 这样做的好处：由于作为上层代理服务器的主机通常具有较高的带宽，这

样通过它去要求数据理论上速度会更快

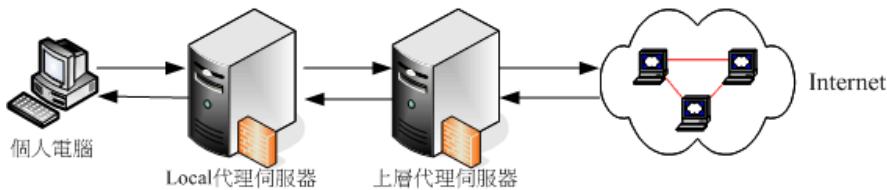


图 17-3 上层代理服务器

2、上层代理服务器最大的好处在于分流

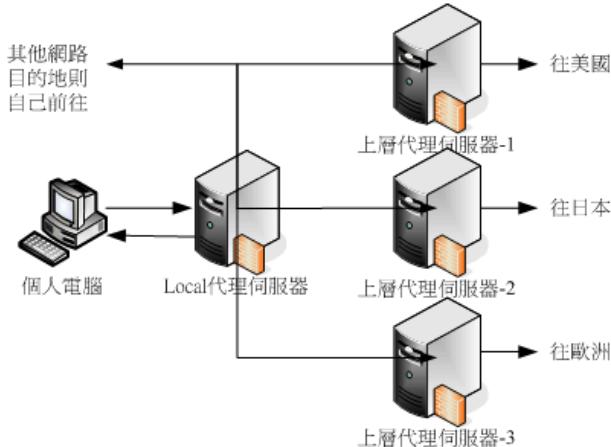


图 17-4 以多台上层代理服务器达到分流的效果示意图

3、由于代理服务器需要管理信任的来源客户端计算机，因此各 ISP 仅能针对自家的用户开放 Proxy 使用权而已

- 由于当用户通过 Proxy 连到因特网时，网络看到的是 Proxy 在获取数据而不是该客户端，因此 Proxy 可能会被客户端过度滥用，同时也有可能会被拿来为非作歹。目前绝大部分的 Proxy 已经停止对外开放了，仅针对自己的网段内的用户提供本项服务而已
- 如果你要自行设置 Proxy 的时候，请记得去申请网络的单位搜寻一下，这样才能有效地设置好你的服务器
- 如果设置错误，上层 Proxy 根本不提供服务，或者上层 Proxy 效率并不好

17.1.4. 代理服务器与 NAT 服务器的差异

1、NAT 服务器的功能

- Linux 的 NAT 功能主要通过数据包过滤的方式，并使用 iptables 的 nat 表格进行 IP 伪装(SNAT)，让客户端自行前往因特网上的任何地方的一种方式
- 主要运作行为是在 OSI 七层协议的第二、三四层
- 由于是通过数据包过滤与伪装，因此客户端可以使用的端口号(第四层)弹性较大

2、Proxy 服务器的功能

- 主要通过 Proxy 的服务程序(daemon)提供网络代理的任务，因此 Proxy 不能进行某些工作，与该服务的程序功能有关
- 如果你的 Proxy 并没有提供邮件或者 FTP 代理，那么你的客户端就无法通过 Proxy 去取得这些网络资源
- 主要工作行为在 OSI 七层协议的应用层部分

3、NAT 服务器是由较底层的网络去进行分析工作，至于通过 NAT 的数据包是什么用的，NAT 不管；而 Proxy 主要是由一个 daemon 的功能实现的，所以必须要符合该 daemon 的需求，才能实现某些功能

17.1.5. 搭建代理服务器的优缺点

1、代理服务器的主要功能：

- 作为 WWW 的网页数据获取代理人，这是最主要的功能
- 作为内部局域网的单点对外防火墙系统

2、由于 Proxy 的这种特性，常被用于大型的企业内部，因为可以实现杜绝内部人员上班时使用非 WWW 以外的网络服务，而且还可以监测用户的资料要求流向与流量

3、Proxy 的主要优点如下

- 节省单点对外的网络带宽，降低网络负载
- 以较短的路径取得网络数据，有网络加速的感觉
 - 假如你指定你的 ISP 提供代理服务器连接到国外，由于 ISP 提供的 Proxy 通常具有较大的对外带宽，因此在国外网站的数据取得上，通常会比你自己的主机连接到国外要快得多
 - 从内部硬盘取得的路径总比对外的因特网要短很多
- 通过上层代理服务器的辅助，达到自动数据分流的效果
- 提供防火墙内部的计算机连上 Internet

4、Proxy 的缺点如下

- 容易被内部局域网的人员滥用
 - 由于因特网上看到取得数据的人是 Proxy 那台主机而不是客户端计算机的 IP
 - 因此可能会让某些内部网络的使用人员开始利用 Proxy 干坏事
 - 为了杜绝该情况，强烈建议多加日志分析的软件
- 需要较高的配置技巧与排错程序
- 可能会取得旧的错误数据

17.2. Proxy 服务器的配置基础

17.2.1. Proxy 所需的 squid 软件及其软件结构

1、以"yum install squid"进行安装

2、squid 的主要配置文件

- /etc/squid/squid.conf：主要配置文件，所有 squid 需要的设置值都是放置在这个文件当中的
- /etc/squid/mime.conf：该文件设置 squid 所支持的 Internet 上面的文件格式，就是所谓的 mime 格式，通常这个文件的默认设置已经能满足我们的需求了

3、其他重要的目录与文件

- /usr/sbin/squid：提供 squid 的主程序
- /var/spool/squid：就是默认的 squid 缓存存储的目录
- /usr/lib64/squid/：提供 squid 额外的控制模块，尤其是影响认证密码方面的程序，都是放置在这个目录下的

17.2.2. CentOS 默认的 squid 设置

1、在默认情况下，CentOS 的 squid 具有以下几个特色

- 仅有本机(localhost,127.0.0.1)来源可以使用这个 squid 功能
- squid 所监听的 Proxy 服务端口在 3128
- 缓存目录所在的位置:/var/spool/squid/，且仅有 100MB 的磁盘高速缓存量
- 处理 squid 程序所需要的基本内存之外，尚提供 8MB 的内存来给热门文件缓存在内存中
- 默认启动 squid 程序的用户为 squid 这个账号(与磁盘高速缓存目录权限有关)

2、帮用户进行监听与传送数据的是 port 3128(TCP)，而 port 3130(UDP)仅是负责与邻近 Proxy 互相沟通彼此缓存数据库的功能，与实际的用户要求无关

3、查看与修改缓存目录(cache_dir): 权限与 SELinux

- squid 如何将缓存存进磁盘？
- squid 是将数据分成一小块一小块，然后分别放置到个别的目录中
- 由于较多的目录可以节省在同一个目录内找好多文件的事件
- 在默认的/var/spool/squid/目录下，squid 又会将它分成两层子目录来存放相关的缓存数据
- cache_dir ufs 设置值

cache_dir ufs /var/spool/squid 100 16 256

- 第一个 100 代表的是磁盘使用量仅用掉该文件系统的 100MB
- 第二个 16 代表第一层此目录共有 16 个
- 第三个 256 代表每层此目录内部再分为 256 个次目录
- **这个设置值可以重复多次出现，所有目录均会被用于存放缓存数据**

- cache_swap_low 与 cache_swap_high 设置值

cache_swap_low 90
cache_swap_high 95

- 为了避免缓存数据塞满磁盘，可以利用上述两个参数来避免
- 当磁盘使用量达 95 时，比较旧的缓存数据将会被删除，当删除到剩下磁盘使用量达 90%时，就停止持续删除的操作

4、squid 使用的内存计算方式

- 除了磁盘容量之外，内存是另一个相当重要的影响 Proxy 效能的因素
- Proxy 会将数据存一份在磁盘高速缓存中，但同时也会将数据暂存在内存当中，以加快未来用户访问同一份数据的速度
- cache_men 设置值
 - cache_men 32MB
 - 并不是指使用多少内存给 squid 使用，而是指额外提供多少内存给 squid 使用

17.2.3. 管理信任来源(如局域网)与目标(如恶意网站):acl 与 http_access 的使用

1、在上面小节的设置中，其实仅有 Proxy 服务器本身可以向自己的 Proxy 要求网页代理，但是重点是开放给局域网来使用这个 Proxy

2、acl 的基本语法

acl <自定义的 acl 名称><要控制的 acl 类型><设置的内容>

- 由于 squid 并不会直接使用 IP 或网络来管控信任目标，而是通过 acl 名称来管理，这个<acl 名称>就必须要设置管理的是来源还是目标(acl 类型)，以及实际的 IP 或网络(设置的内容)

2、管理是否能使用 Proxy 的信任客户端方式(**例句中绿色斜体为关键字**)

- 由于因特网主要有使用 IP 或主机名来作为连接方式，因此信任用户的来源至少包含以下几种
- **src ip-address/netmask**
 - 主要控制来源的 IP 地址
 - **acl vbirdlan src 192.168.1.0/24 192.168.100.0/24**
- **src addr1-addr2/netmask**
 - 主要控制一段范围来源的 IP 地址
 - **acl vbirdlan2 src 192.168.1.100-192.168.1.200/24**
- **srcdomain .domain.name**
 - 如果来源用户的 IP 一直变，使用的是 DDNS 的方式来更新主机名和 IP，此时可以用主机名来开放，例如来源是.ksu.edu.tw 的来源用户开放使用权
 - **acl vbirdsu srcdomain .ksu.edu.tw**

4、管理是否让 Proxy 帮忙代理到该目标去获取数据

- 除了管理来源用户之外，还能够管理是否让 Proxy 服务器到某些目标去获取数据
- 在默认设置中，Proxy 仅管理可以向外取得 port21、80、440 等端口的目标网站，不是这些端口的就无法帮忙代理取得
- 基本的管理方法有如下几种
- **dst ip-addr/netmask**
 - 控制不能去的目标网站 IP
 - **acl dropip dst 120.114.150.21/32**
- **dstdomain .domain.name**
 - 控制不能去的目标网站的主机名
 - **acl dropfb dstdomain .facebook.com**
- **url_regex [-i] ^http://url**
 - 使用正则表达式来处理网址的方式
 - 这种方式的网址必须要完整地输入正则表达式的开始到结尾才行
- **urlpath_regex [-i] \.gif\$**
 - 与上一个 acl 非常相似，上一个需要填写完整的网址数据，这里则是根据网址列的部分比对来处置
- **acl sexurl urlpath_regex /sexy.*\.jpg\$**
- **acl dropdomain dstdomain "/etc/squid/dropdomain.txt"**

5、以 http_access 调整管理信任来源与管理目标的顺序

- 设置好 acl 之后，接下来就要看是否放行
- 放行与否与 http_access 有关，基本上就是拒绝(deny)与允许(allow)两个控件目
 - **http_access deny [acl 名称]**
 - **http_access allow [acl 名称]**

- 顺序比对，跟 **iptables** 规则类似，先将要拒绝的写上去，然后在写要放行的数据

17.2.4. 其他额外的功能项目

1、不要进行某些网页的缓存操作

- Proxy 的缓存通常记录比较少变动的数据，如果是讨论区或者是程控类的数据库形态网页，那就没有缓存的必要，因为数据一直变动
- 例如，.php 结尾通常是讨论区之类变动性的数据
 - **acl denyphp urlpath_regex \.php\$**
 - **cache deny denyphp** <==denyphh 只是一个 acl 名称

2、磁盘中缓存的存在时间

- **refresh_pattern** <regex> <最小时间> <百分比> <最大时间>
- **regex:** 使用的是正则表达式来分析网址的数据
- 最小时间：单位是分钟，当取得这个数据的时间超过这个设置值，该数据会被判定为旧数据
- 百分比：这个与"最大时间"有关，当该数据被获取到缓存后，经过最大时间的多少百分比时，该数据就会被重新获取
- 最大时间：这个数据在缓存内的最长时间

3、主机名与管理员的 E-mail 定义

- 通过 **visible_hostname** 来指定主机名
- 通过 **cache_mgr** 指定管理员 E-mail，当客户端使用 squid 出现任何错误时，屏幕上都会出现管理员的 E-mail 让用户联系

17.2.5. 安全性设置：防火墙、SELinux 与黑名单文件

1、防火墙需要放行 TCP 的 port 3128 端口

2、SELinux 的注意事项

- 配置文件(/etc/squid/内的数据)类型是：squid_conf_t
- 缓存目录类型是：squid_cache_t
- 上层类型(/var/spool/)是：var_t

3、建立黑名单配置文件

- **acl dropdomain dstdomain "/etc/squid/dropdomain.txt"**
- **systemctl reload squid.service** <==reload 比 restart 快

17.3. 客户端的使用与测试

1、浏览器的设置：Firefox&IE

17.4. 服务器的其他应用设定

17.4.1. 上层 Proxy 与获取数据分流的设定

1、<未完成>

17.4.2. Proxy 服务放在 NAT 服务器上：透明代理 (Transparent Proxy)

1、如何强制用户一定要使用 Proxy

- 在对外防火墙服务器(NAT)上面安装 Proxy
- 在 Proxy 上启动 transparent 功能
- NAT 服务器上加一条 80 转 3128 的规则
- 这样一来，所有往 port80 的数据包就会被 NAT 转向 port 3128，而 port 3128 就是 Proxy
- 这样可以强制用户使用 Proxy，并且浏览器不需要任何设置

17.4.3. Proxy 的认证设置

1、一般来说，Proxy 只会开放内部网络的人们来使用，但是如果我在 Internet 也想用这台 Proxy，那该怎么办

2、squid 官方软件已经给予了认证的设置功能，即我们可以通过认证来简单地输入账号密码

- 使用 squid 提供的 ncsa_auth 认证模块，这个模块会利用 apache(WWW 服务)提供的账号密码建立命令(htpasswd)所制作的密码文件作为验证依据

3、步骤

- yum install httpd <==取得/usr/bin/htpasswd 这个命令
- vim /etc/squid/squid.conf
 - auth_param basic program /usr/lib64/squid/basic_ncsa_auth /etc/squid/squid_user.txt <==通过 basic_ncsa_auth 读取密码*
 - auth_param basic children 5 <==启动 5 个程序来管理验证的需求*
 - auth_param basic realm Welcome to Liuye's proxy web server<==欢迎界面*
 - acl squid_user proxy_auth REQUIRED <==建立 acl 名称*
 - http_access allow squid_user <==将这句放在规则较后面的位置*
- 其中 REQUIRED 是指定任何在密码文件内的用户都能够使用验证

17.4.4. 末端日志分析：SARG

1、Squid Analysis Report Generator(SARG,Squid 分析报告制作者)，它的原理相当简单，就是讲 logfile 拿出来，然后进行解析，依据不同的时间、网站、与热门网站等来进行数据的输出，输出结果非常详细

Chapter 18. 网络驱动器设备：iSCSI 服务

18.1. 网络文件系统还是网络驱动器

1、作为服务器的系统通常是需要存储设备的，而存储设备除了可以使用系统内置的磁盘之外，如果内置的磁盘容量不够大，而且也没有额外的磁盘插槽(SATA 或 IDE)可用时，那么常见解决的方案就是增加 NAS 或外接式存储设备，或者 SAN(局域网存储)

18.1.1. NAS 与 SAN

1、磁盘阵列的位置

- 主机内部有磁盘阵列控制卡，可以自行管理磁盘阵列，不过想要提供磁盘阵列的容量，需要通过额外的网络服务才行
- 外接式磁盘阵列设备，就是单纯的磁盘阵列设备，必须通过某些接口连接到主机上，主机也要安装适当的驱动程序后，才能捕捉到这个设备所提供的磁盘容量

2、NAS(Network Attached Storage，网络附加存储服务器)

- NAS 就是一台定制化好的主机，只要将 NAS 连接上网络，那么在网络上面的其他主机就能够访问 NAS 上的数据
- 简单地说，NAS 就是一台 File Server
- 由于 NAS 连接在网络上，如果网络上有某个用户大量访问 NAS 上的数据，容易造成网络停顿的问题
- NAS 通常支持 TCP/IP，并会提供 NFS, SAMBA, FTP 等常见的通信协议来提供客户端取得文件系统
- 由于 NAS 会包括许多的配置接口，通常是利用 Web 接口来控制磁盘阵列的设置情况、提供 IP 或其他相关网络设置，以及是否提供某些特定的服务，因为具有较为亲和的操作与控制接口，对于非 IT 人员来说，控制较为容易

3、SAN(Storage Area Networks，存储局域网)

- SAN 可以通过某些特殊的接口或信道来提供局域网内的所有机器进行磁盘访问，SAN 是提供"磁盘(block device)"给主机用，而不是像 NAS 提供"网络协议的文件系统"
- 挂载使用 SAN 的主机会多出一个大磁盘，并可针对 SAN 提供的磁盘进行分区与格式化操作
- SAN 最大的目的就是提供磁盘给服务器主机使用，因此 NAS 可以使用 SAN

18.1.2. iSCSI 接口

1、iSCSI 主要是通过 TCP/IP 的技术，将存储设备端通过 iSCSI target(iSCSI 标的)功能，做成可以提供磁盘的服务器端，再通过 iSCSI initiator(iSCSI 初始化用户)功能，做成能够挂载使用 iSCSI target 的客户端，如此便能通过 iSCSI 协议来进行磁盘的应用了

2、iSCSI 这个架构主要将存储设备与使用的主机分为两个部分

- iSCSI target：存储设备端，存放磁盘或 RAID 的设备，目前也能够将 Linux 主机仿真成 iSCSI target，目的在于提供其他主机使用的磁盘

- iSCSI initiator: 就是能够使用 target 的客户端，通常是服务器

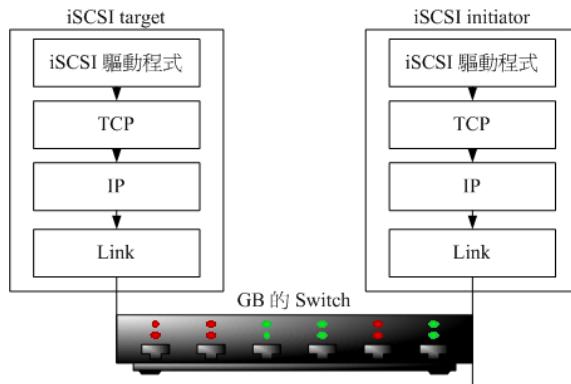


图 18-1 iSCSI 与 TCP/IP 相关性

18.1.3. 各组件相关性

1、如何取得磁盘或者文件系统

- 直接访问(direct-attached storage): 例如本机上面的磁盘，就是直接访问设备
- 通过存储局域网(SAN): 来自局域网内的其他存储设备提供的磁盘
- 网络文件系统(NAS): 来自 NAS 提供的文件系统，只能立即使用，不可进行格式化

18.2. iSCSI target 的设置

- 1、CentOS 利用的是 tgt 这个软件
- 2、CentOS7 以后用的是 targetcli 工具

18.2.1. 所需软件与软件结构

1、所需软件

- targetcli: 用来将 Linux 系统仿真成为 iSCSI target 的功能
 - iscsi-initiator-utils: 挂载来自 target 的磁盘到 Linux 本机上
- <未完成>: 软件结构发生了变化，书上的都不对了

18.2.2. iSCSI target 的实际设置

1、iSCSI 就是通过一个网络接口，将既有的磁盘共享出去，以下是可共享的磁盘类型

- 使用 dd 命令所建立的大型文件可供仿真为磁盘(无需预先格式化)
- 使用单一分区(partition)共享为磁盘
- 使用单一完整的磁盘(无需预先分区)
- 使用磁盘阵列共享(其实与单一磁盘相同方式)
- 使用软件磁盘阵列(software RAID)共享成单一磁盘
- 使用 LVM 的 LV 设备共享为磁盘

2、建立所需要的磁盘设备

- 建立一个名为/srv/iscsi/disk1.img 的 500MB 文件
- 使用/dev/sda10 提供 2GB 作为共享

Chapter 19. 主机名控制者：DNS 服务器

19.1. 什么是 DNS

19.1.1. 用网络主机名取得 IP 的历史渊源

1、单一文件处理上网的年代：/etc/hosts

- 利用某些特定的文件将主机名与 IP 做一个对应，如此一来，我们就可以通过主机名来取得该主机的 IP 了
- 但是主机名与 IP 的对应无法自动在所有计算机内更新，且要将主机名加入该文件仅能向 INTERNIC 注册，若 IP 数量太多时，该文件会过大，也就更不利于其他主机同步化了

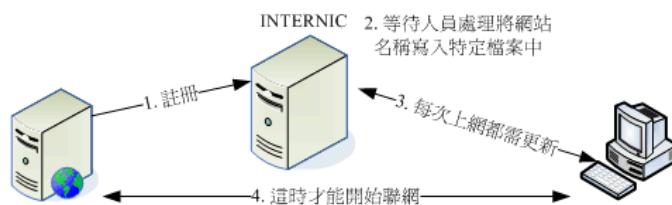


图 19-1 早期通过单一文件进行网络连接的示意图

- 在私有网络内部，最好将所有的私有 IP 与主机名对应都写入这个文件中

19.1.2. DNS 的主机名对应 IP 的查询流程

1、DNS 的阶层架构与 TLD

- 在整个 DNS 系统的最上方一定是"."(小数点)这个 DNS 服务器，称为 root，最早在它下面管理的就只有.com、.edu、.gov、.mil、.org、.net 这种特殊区域以及以国家为分类的第二层的主机名，这两者称为 Top Level Domains(TLDs)
- 一般顶级域名(Generic TLDs, gTLD): 例如.com、.org、.gov 等
- 地区顶级域名(Country Code TLDs, ccTLD): 例如.uk、.jp、.cn 等

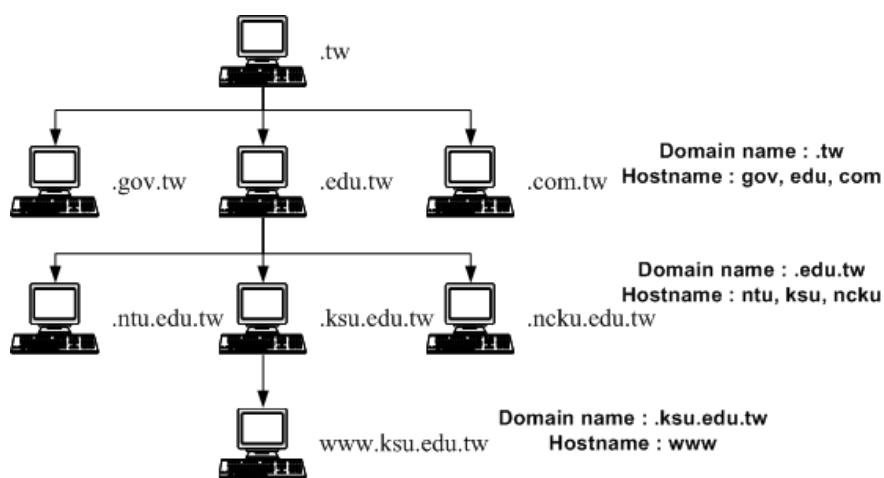


图 19-2 从顶级域名到昆山科大之间的 DNS 阶层示意图

- 最早 root 管理六大领域域名，如下表

表格 19-1 常见域名及其代表意义

| 名称 | 代表意义 |
|-----|----------|
| com | 公司、行号、企业 |

| | |
|-----|-------|
| org | 组织、机构 |
| edu | 教育单位 |
| gov | 政府单位 |
| net | 网络、通信 |
| mil | 军事单位 |

- 由于因特网发展非常快，除了上述六大类别之外，还有.asia、.info、.jobs 等域名的开放
- 此外，为了让国家也能够有自己的顶级域名，就有所谓的 ccTLD 了，如此一来，如果有 domain name 的需求，则只要向自己的国家申请即可，不需要再到最上层去申请

2、授权与分层负责

- 我们不能执行设置 TLD，必须向上层 ISP 申请域名的授权才行
- 每个国家之下记录的主要区域，基本与 root 管理的六大类类似
- **每一个上层的 DNS 服务器所记录的信息，其实只有其下一层的主机名而已**
 - 例如图 19-2，.tw 只记录下面那一层的这个主要的 domain 的主机而已
 - 至于 edu.tw 下面的 ksu.edu.tw 这台机器，那就直接授权给 edu.tw 那台机器去管理
- 这样设置的好处：每台机器管理的只有下一层的 hostname 对应的 IP，所以减少了管理上的困扰，如果下层 Client 端如果有问题，只要询问上一层 DNS Server 即可，不需要跨越上层，排错也会比较简单

3、通过 DNS 查询主机名 IP 的流程

- DNS 是以类似树形目录的形态来进行主机名的管理的，所以每一台 DNS 服务器都仅管理自己的下一层主机名的转译，至于下层的下层，则授权给下层的 DNS 主机来管理

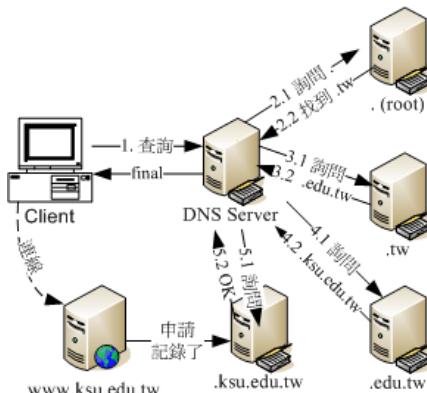


图 19-3 通过 DNS 系统查询主机名解析的流程

- 当你在浏览器地址输入 <http://www.ksu.edu.tw> 时，计算机会依据相关设置(在 Linux 下面就是利用/etc/resolv.conf 这个文件)所提供的 DNS(以 168.95.1.1)的 IP 去进行连接查询
 - 1) 收到用户的查询要求，先查看本身有没有记录，若无则向.(root)查询
 - 2) 向最顶层的.(root)查询
 - 由于.(root)只记录了.tw 的信息(因为台湾地区只有.tw 向.(root)注册)
 - 从而转向.tw 查询
 - 3) 向第二层的.tw 服务器查询

- 这台机器(.tw)管理的仅仅有.edu.tw、.com.tw、gov.tw 等
 - 转向.edu.tw 这个区域的主机查询
- 4) 向第三层的.edu.tw 服务器查询
 - 查询到.ksu.edu.tw 的 IP
 - 转向.ksu.edu.tw 查询
 - 5) 向第四层的.ksu.edu.tw 服务器查询
 - 找到了 www.ksu.edu.tw
 - 6) 记录缓存并回报用户
 - DNS(168.95.1.1)会先记录一份查询结果在自己的缓存中，以方便响应下一次的相同要求
- 整个分层查询的流程就是这样，总要先经过.(root)来向下一层进行查询，这样分层的好处如下
- 主机名修改的仅需要更改自己的 DNS 即可，不需要通知其他人
 - 只要你的主机名是经过上层合法的 DNS 服务器设置的，那么就可以在 Internet 上被查到
 - DNS 服务器对主机名解析结果的缓存时间
 - 在缓存内的答案是有时间性的，通常是数分钟到三天之内
 - 因此当修改了一个 domain name 后，可能要 2-3 天才能全面启用
 - 可持续向下授权(子域名授权)
 - **dig +trace www.ksu.edu.tw** <==显示分层查询的过程
- 4、DNS 使用的 port number
- DNS 使用的 port 是 53，在/etc/services 这个文件搜寻 domain 关键词就可以查到 53 这个 port
- 通常 DNS 是以 UDP 这个较快速的数据传输协议来查询，如果没有查到完整的信息，就会再次以 TCP 这个协议来重新查询，因此防火墙要放行 TCP\UDP 的 port53

19.1.3. 合法 DNS 的关键：申请区域查询授权

1、DNS 服务器的架设还有合法与不合法之分，而不像其他服务器一样，架设好之后别人就查得到

2、向上曾区域注册取得合法的区域查询授权

- 申请一个合法的主机名就需要注册，注册就需要花钱
- 注册取得的数据有两种
 - FQDN(主机名)：只需要主机名，详细的设置数据就由 ISP 帮我们搞定
 - 申请区域查询权
 - 以.ksu.edu.tw 为例
 - .ksu.edu.tw 必须要向.edu.tw 那台主机注册申请区域授权
 - 未来有任何.ksu.edu.tw 的要求时，.edu.tw 都会转向.ksu.edu.cn
 - 此时我们就需要架设 DNS 服务器来设置.ksu.edu.tw 相关的主机名对应才行
- 架设 DNS，而且是可以连上 Internet 上面的 DNS，就必须通过上层 DNS 服务器的授权才行
- **让你的主机名对应 IP 且让其他计算机可以查询到，可以有如下两种方式**
 - 上层 DNS 授权区域查询权，让你自己设置 DNS 服务器

- 直接请上层 DNS 服务器来帮你设置主机名对应
- 3、拥有区域查询权后，所有的主机名信息都以自己为准，与上层无关
- DNS 系统记录的信息非常多，重点有两个
 - 记录服务器所在的 NS(NameServer)标志
 - 记录主机名对应的 A(Address)标志

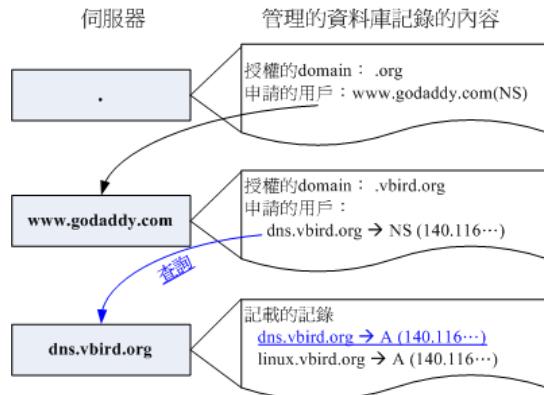


图 19-4 记录的授权主机名与实际 A 记录的差异

- 架设的 DNS 主机在其上层 DNS 主机中记录的是 NS 标志

19.1.4. 主机名交由 ISP 代管还是自己设置 DNS 服务器

- 1、无论如何你只有两个选择
 - 请上层 DNS 帮你设置好 Hostname 与 IP
 - 请上层 DNS 将某个 domain name 段授权给你作为 DNS 的主要管理区域
- 2、需要架设 DNS 的时机
 - 你所负责需要连上 Internet 的主机数量庞大，例如负责公司几十台的网络 Server，而这些 Server 都是挂载在你公司的区域之下
 - 你可能需要时常修改你的 Server 的名字，或者你的 Server 有随时增加的可能性与变动性
- 3、不需要架设 DNS 的时机
 - 网络主机数量很少
 - 你可以直接请上层 DNS 主机管理员帮你设置好 Hostname 的对应
 - 你对于 DNS 的认知不足，如果架设反而容易造成网络不通的情况
 - 架设 DNS 费用很高

19.1.5. DNS 数据库的记录：正解、反解、Zone 的意义

- 1、记录的东西可以称为数据库，在数据库里针对每个要解析的域(domain)，就称为一个区域(zone)
- 2、概念
 - 正解：从主机名查询到 IP 的流程
 - 反解：从 IP 反解析到主机名的流程
 - 无论正解还是反解，每个域记录的就是一个区域(zone)
 - 举例来说
 - 昆山科大 DNS 服务器管理的就是*.ksu.edu.tw 这个域的查询权，任何想知道*.ksu.edu.tw 主机名的 IP 都得向昆山科大的 DNS 服务器查询，此时.ksu.edu.tw 就是一个“正解的区域”

- 昆山科大有几个 Class C 的子域，例如 120.114.140.0/24，如果这 254 个可用 IP 都要设置主机名，那么这个 120.114.140.0/24 就是一个“反解的区域”

3、正解的设置权以及 DNS 正解 Zone 记录的标志

- 只要改域没有人使用，那谁先抢到了，就能够使用了
- 正解的 Zone 通常有以下几种标志
 - SOA：就是开始验证(Start of Authority)的缩写
 - NS：就是名称服务器(Name Server)的缩写，后面记录的数据时 DNS 服务器
 - A：就是地址(Address)的缩写，后面记录的是 IP 的对应

4、反解的设置权以及 DNS 反解 Zone 记录的标志

- 反解主要是由 IP 找到主机名，重点是 IP 所有人是谁
- IP 都是 INTERNIC 发放给各家 ISP 的，而且 IP 不能乱设置(路由问题)
- **因此能够设置反解的就只有 IP 的拥有人，亦即 ISP 才有权利设置反解**
- 除非你取得是整个 Class C 以上等级的 IP 网段，那你的 ISP 才可能给你 IP 反解授权，否则，若有反解的需求，就需要向你的直属上层 ISP 申请才行
- 反解的 Zone 记录的主要信息如下：
 - NS
 - SOA
 - PTR：就是指向(Pointer)的缩写，后面记录的数据就是反解到的主机名

5、每台 DNS 都需要的正解 Zone：hint

- 一个正解或一个反解就可以称为一个 Zone 了
- **“.”这个 Zone 是最重要的**
 - 从图 19-3 可以知道
 - 当 DNS 服务器在自己的数据找不到所需的信息时，一定会去找“.”
 - 因此就必须具有记录“.”在哪里的记录 Zone 才行
 - 这个记录“.”的 Zone 类型，就被称为 hint 类型，这个几乎是每个 DNS 服务器都需要知道的 Zone
- **一台简单的正解 DNS 服务器，基本就要有两个 Zone 才行，一个是 hint，一个是关于自己域的正解 Zone**

6、正反解是否一定要成对

- 正反解不需要成对
- 在很多情况下，常常会只有正解的设置需求
- 事实上，需要正反解成对需求的大概仅有 Mail Server，由于目前网络带宽经常被垃圾邮件、广告邮件占光，所以 Internet 的社会对于合法的 Mail Server 规定也就越来越严格

19.1.6. DNS 数据库的类型：hint、Master/Slave 架构

1、现在注册域名，那么 ISP 都会要求你填写两台 DNS 服务器的 IP，因为需要作为备份之用

- 如果有两台以上的 DNS 服务器，那么网络上会搜寻到哪一台是不确定的，因为是随机的
- 这两台 DNS 服务器的内容必须一模一样，如果不同步将会造成用户无法取得正确数据的问题

- 为了解决这个问题，因此在.(root)这个 hint 类型的数据库文件外，还有两种基本类型
 - Master(主人，主要)数据库
 - Slave(奴隶，次要)数据库

2、Master

- 这种类型的 DNS 数据库中，里面所有的主机名相关信息，都需要管理员自己手动去修改与设置，设置完毕后还需要重新启动 DNS 服务去读取正确的数据库内容，才算完成数据库更新
- 一般来说，我们说的 DNS 假设，就是指这种数据库的类型
- 同时这种类型的数据库，还能够提供数据库内容给 Slave 的 DNS 服务器

3、Slave

- 通常不会只有一台 DNS 服务器，如果每台 DNS 都使用 Master 数据库类型，当有用户想我们要求修改或添加、删除数据时，一笔数据就需要做三次
- Slave 必须与 Master 相互搭配
- 假如有三台主机提供 DNS 服务，且三台内容相同，那么只需要指定一台服务器为 Master，其他两台为该 Master 的 Slave 服务器，那么当要修改的一笔名称对应时，我们只要手动更改 Master 那台机器的配置文件，然后重新启动 BIND 这个服务后，其他两台 Slave 就会自动被通知更新了

4、Master/Slave 的查询优先权

- 不论是 Master 还是 Slave 服务器，都必须可以同时提供 DNS 服务才行，因为在 DNS 系统中，域名的查询是"先进先出"的状态
- 每一台 DNS 服务器的数据库内容需要完全一致，否则就会造成客户端找到的 IP 是错误的

5、Master/Slave 数据的同步化过程

- Slave 需要更新来自 Master 的数据，所以当然 Slave 在设置之初就需要存在 Master 才行
- 不论 Master 还是 Slave 数据库，都会有一个代表该数据库新旧的"序号"，这个序号数值的大小，是会影响是否要更新的操作
- 更新的方式主要有以下两种
 - Master 主动告知：在 Master 修改了数据库内容，并加大数据库序号后，重新启动 DNS 服务，那 Master 会主动告知 Slave 来更新数据库，此时能实现数据同步
 - Slave 主动提出要求：Slave 会定时向 Master 查看数据库的序号，当发现 Master 数据库序号比 Slave 序号更大时，那么 Slave 就会开始更新，如果序号不变，就判断数据库没有变动，不会进行同步更新

19.2. Client 端的设置

19.2.1. 相关配置文件

1、问题：

- 从主机名对应到 IP 有两种方法，/etc/hosts 或者通过 DNS 架构
- 那么这两种方法分别使用什么配置文件
- 可否同时存在

- 若可以同时存在，哪种优先

2、如下几个配置文件

- `/etc/hosts`: 最早的 Hostname 对应 IP 的文件
- `/etc/resolv.conf`: 就是 ISP 的 DNS 服务器 IP 记录处
- `/etc/nsswitch.conf`: 决定先使用`/etc/hosts` 还是`/etc/resolv.conf` 的设置

3、一般而言，默认的 Linux 主机与 IP 的对应解析都以`/etc/hosts` 优先

- `vim /etc/nsswitch.conf <==找到 hosts: 项目`
- `hosts: files dns`
- `files:` 就是使用`/etc/hosts`
- `dns:` 使用`/etc/resolv.conf`

4、`/etc/resolv.conf`

- DNS 服务器的 IP 可以设置多个，为什么要多个(避免 DNS 服务器宕机找不到 IP)，有点类似 DNS 备份功能
- 在正常情况下，永远只有第一台 DNS 服务器会被用来查询
- **为什么`/etc/resolv.conf` 内容会被自动修改**
 - 当使用 DHCP 时，系统会主动使用 DHCP 服务器传来的数据进行系统配置文件的修订
 - 如果不想使用 DHCP 传来的服务器设置值，可以在`/etc/sysconfig/network-scripts/ifcfg-eth0` 等相关文件内增加如下一行，然后重新启动即可
`PEERDNS=no`

19.2.2. DNS 的正解、反解查询命令：`host`、`nslookup`、`dig`

1、`<host>`

- `host [-a] FQDN [server]`
- `host -l domain [server]`
- `-a`: 代表列出该主机所有的相关信息，包括 IP、TTL 与排错信息等
- `-l`: 若后面接的那个 `domian` 设置允许 `allow-transfer` 时，则列出该 `domain` 所管理的所有主机名对应数据
- `server`: 这个参数可有可无，当想要利用非`/etc/resolv.conf` 内的 DNS 主机来查询主机名与 IP 的对应时，就可以利用这个参数了

2、`<nslookup>`

- `nslookup [FQDN] [server]`
- `nslookup`

3、`<dig>`

- `dig [option] FQDN [@server]`
- `@server`: 如果不以`/etc/resolv.conf` 的设置来作为 DNS 查询，可在此填入其他的 IP
- `options`: 主要参数有以下几种
 - `+trace`: 从`".`开始追踪
 - `-t type`: 查询的数据主要有 MX, NS, SOA 等类型
 - `-x`: 查询反解信息，非常重要
- 查询结果介绍
 - 整个显示出的信息包括以下几个部分
 - `QUESTION(问题)`: 显示所要查询的内容

- ANSWER(回答): 依据刚刚的 QUESTION 去查询所得到的结果
 - AUTHORITY(验证): 查询结果是由哪台服务器提供的(好像没有了???)
- 例子
- dig linux.vbird.org
 - dig -x 120.114.100.20

19.2.3. 查询域管理者相关信息: whois

- 1、安装: yum install whois
- 2、whois 用于查询这个域是谁管的

19.3. DNS 服务器的软件、种类与 Caching only DNS 服务器设置

19.3.1. 搭建 DNS 所需要的软件

1、我们要使用的 DNS 软件就是使用伯克莱大学发展出来的 BIND(Berkeley Internet Name Domain)

- yum install bind-libs
- yum install bind-utils
- yum install bind
- yum install bind-chroot
- rpm -qa | grep '^bind'
- chroot 代表的是"change to root(根目录)", root 代表的是根目录
 - 早期 BIND 默认将程序启动在 var/named 当中, 但是该程序可以在根目录下的其他目录到处转移, 若 BIND 程序有问题, 则该程序会造成整个系统的危害
 - 为了避免这个问题, **我们将目录指定为 BIND 程序的根目录, 由于已经是根目录, 所以 BIND 便不能离开该目录**
 - CentOS 默认将 BIND 锁在 /var/named/chroot 目录中

19.3.2. BIND 的默认路径设置与 chroot

1、搭建好 BIND 需要设置的两个主要数据

- BIND 本身的配置文件(/etc/named.conf): 主要规范主机的设置、zone file 的存在、权限的设置等
- 正反解数据库文件(zone file): 记录主机名与 IP 的对应等

2、一般来说, CentOS 的默认目录如下

- /etc/named.conf: 配置文件
- /etc/sysconfig/named: 是否启动 chroot 及额外参数
- /var/named/: 数据库文件默认放置的目录
- /var/run/named: named 这支程序执行时默认放置 pid-file 在此目录内

3、一般来说, 目前主要的 distribution 都已经自动地将 BIND 相关程序给 chroot 了

- chroot 指定的目录记录在 /etc/sysconfig/named
- <未完成>: 该文件内容与书上不一致

19.3.3. 单纯的 cache-only DNS 服务器与 forwarding 功能

1、什么是 cache-only 与 forwarding DNS 服务器

- 有个只需要"."这个 zone file 的简单 DNS 服务器，我们称这种没有公开的 DNS 数据库的服务器为 cache-only(唯高速缓存) DNS Server
 - 这个 DNS Server 只有缓存搜寻结果的功能
 - 它本身并没有主机名与 IP 正反解的配置文件，完全是由对外的查询来提供它的数据源
 - 不论谁来查询数据，这台 DNS 一律开始从自己的缓存以及"."找起
- 如果连"."都不想要，那就需要指定一个上层 DNS 服务器作为 forwarding 目标
 - 将原本自己要往"."查询的任务，丢给上层 DNS 服务器去处理即可
 - 即便 forwarding DNS 具有"."这个 zone file，还是会将查询权委托请求上层 DNS 查询

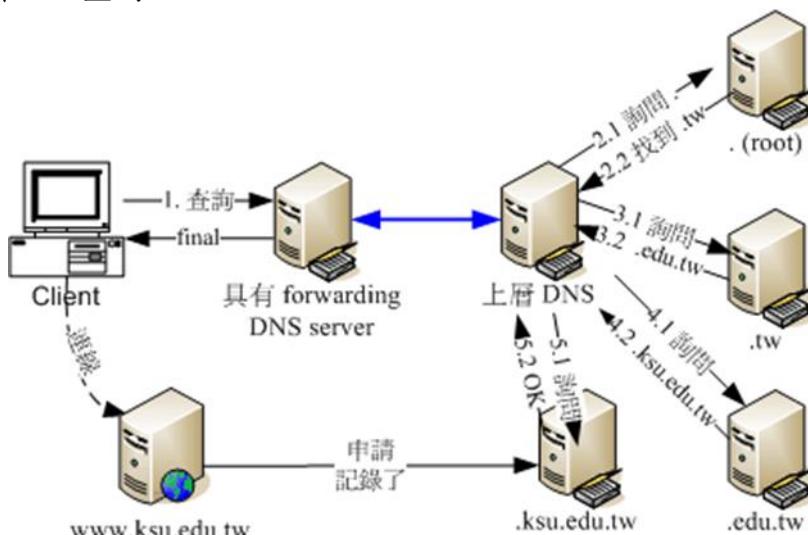


图 19-5 具有 forwarding 功能的 DNS 服务器查询方式示意图

2、什么时候有搭建 cache-only DNS 的需求

- 在某些公司，为了预防员工利用公司网络资源做自己的事情，所以都会针对 Internet 的连接作比较严格的限制，连 port 53 这个 DNS 会用到的 port 也可能被挡在防火墙外，此时，可以在防火墙那台机器上面，加装一个 cache-only DNS 服务器
- 即利用自己的防火墙主机上的 DNS 服务区帮你的 Client 端解析 hostname<-->IP，因为防火墙主机可以设置放行自己的 DNS 功能，而 Client 端就设置该防火墙 IP 为 DNS 服务器的 IP 即可，这样就可以取得主机名与 IP 的转译了
- 通常搭建 cache-only DNS 服务器大都是为了系统安全

3、实际设置 cache-only DNS Server

- 由于不需要设置正反解的 Zone，只需要"."的 Zone 支持即可
- 所以只需要设置一个文件(named.conf 主配置文件)
- 另外，cache-only 只要加上 forwarders 的设置即可指定 forwarding 的数据
- 详细步骤如下

1) 编辑主要配置文件/etc/named.conf

- 以下为参数解释

- `listen-on port 53{any;};`: 监听在这台主机系统上面的那个网络接口，默认是监听在 `localhost`，亦即只有本机可以对 DNS 服务器进行查询，因此要改成 `any`。注意，由于可以监听多个接口，因此 `any` 后面必须有分号
 - `directory "/var/named"`: 如果此文件下面有规范到正/反解的 zone file 文件，该文件名默认应该存储到哪个目录，默认放到`/var/named` 下面，由于 `chroot` 的关系，最终这些数据库文件会被主动链接到`/var/named/chroot/var/named/`这个目录
 - `dump-file`、`statistics-file`、`memstatistics-file`: 与 named 这个服务有关的许多统计信息
 - `allow-query{any;};`: 针对客户端的设置，表示到底谁可以对该 DNS 服务器提出查询请求的意思。**原本文件的内容**默认是针对 `localhost` 开放，这里改成对所有用户开放。**默认的 DNS** 就是对所有用户开放，因此这个设置值可以不用写(**注意原本文件内容与默认的 DNS 的区别**)
 - `forward only;`: 这个设置可以让 DNS 服务仅进行 forward，即便有`".`这个 zone file，也不会使用`".`的数据，只会将查询权上交给上层 DNS 服务器，这个是 cache only DNS 最常见的设置
 - `forwarders{10.3.9.6;};`: 可以设置多个 DNS 服务器，避免其中一台宕机导致服务停止，每一个 forwarder 服务器的 IP 都需要有`";`来结尾，另外大括号外也要有`";`
- 2) 启动 named 并查看服务器的端口
 - `systemctl restart named.service`
 - `netstat -tunlp | grep named`
 - 3) 检查`/var/log/messages`的日志信息(极重要)
 - 4) 测试
 - 如果你的 DNS 服务器具有连上因特网的功能，那么通过
`dig www.baidu.com @127.0.0.1`
 - 如果找到 `www.baidu.com` 的 IP，并且最先显示 "`SERVER:127.0.0.1#53(127.0.0.1)`" 字样，就代表成功了

4、forwarders 的好处与问题分析

- 利用 forwarders 的功能来提高效率的理论
 - 当很多下层 DNS 服务器都使用 forwarders 时，那么那个被设置为 forwarder 的主机，由于会记录很多的查询信息记录
 - 因此对那些下层的 DNS 服务器而言，查询速度会快很多，亦即节省很多查询时间，因为 forwarder 服务器里面有较多缓存记录
 - 因此包括 forwarder 本身，以及所有向这台 forwarder 要求数据的 DNS 服务器，都能减少往`".`查询的机会，因此速度当然加快
- 利用 forwarders 反而会使整体的效率降低
 - 当 DNS 本身的业务量就很繁忙时，那么你的 cache-only DNS 服务器还要向他要求数据，因为它原本的数据传输量就太大了，带宽方面可能负荷过大，而太多的下层 DNS 还向它要求数据，所以它的查询速度会变慢
 - 而你的 cache-only Server 又是向它提出要求的，因此自然两边的查询速度都会同步下降

- 如果上层 DNS 速度很快的话，那么它被设置为 forwarder 时，或许真的可以提高不少效率

19.4. DNS 服务器的详细设置

1、架设 DNS 服务器的细节

- DNS 服务器的架设需要上层 DNS 的授权才可以成为合法的 DNS 服务器(否则只是练习)
 - 没有注册，就代表在上层的 DNS 服务器中是没有这台 DNS 服务器的记录的，即本台 DNS 服务器是没有一个合法域名，因此不具备分配主机名的能力
 - 但是没有注册，只要该 DNS 可以正常联网，那么其他主机仍然可以通过这台 DNS 来查询“合法主机名”的 IP，这里的合法主机名必然是在其他 DNS 服务器进行注册过的主机名
- 配置文件的位置：目前 BIND 程序已进行 chroot，相关目录参考 /etc/sysconfig/named
- named 主要配置文件：/etc/named.conf
- 每个正、反解区域都需要一个数据库文件，而文件名则是由 /etc/named.conf 所设置
- 当 DNS 查询时，若本身没有数据库文件，则前往 root("." 或 forwarders 服务器查询
- named 能否成功启动务必要查询 /var/log/messages 内的信息(其实 systemctl restart named.service 时若失败则会提供日志查询的命令的)

19.4.1. 正解文件记录的数据(Resource Record, RR)

1、正解文件资源记录(Resource Record, RR)格式

- 我们可以发现，dig 命令输出结果的格式几乎是固定的

| | | | | |
|----------|-----------|----|-------------|------------|
| [domain] | [ttl] | IN | [[RR type]] | [RR data]] |
| [待查数据] | [暂存时间(秒)] | IN | [资源类型] | [资源内容] |

 - RR type 与 RR data 是互相有关联性的
 - A 后面接 IP
 - NS 后面接主机名
 - 此外，在 domain 部分，若可能的话，请尽量使用 FQDN，即主机名结尾加一个小数点(".")，就被称为 FQDN
 - ttl: time to live 的缩写，这笔记录被其他 DNS 服务器查询到后，这个记录会在对方 DNS 服务器的缓存中，保持多少秒钟的意思，由于 ttl 可由特定参数统一管理，因此在 RR 记录格式中，通常这个 ttl 字段是可以忽略的
- 正解文件的 RR 记录格式汇整如下

| [domian] | IN | [[RR type]] | [RR data]] |
|----------|----|-------------|---------------|
| 主机名. | IN | A | IPv4 的 IP 地址 |
| 主机名. | IN | AAAA | IPv6 的 IP 地址 |
| 域名. | IN | NS | 管理这个域名的服务器主机名 |
| 域名. | IN | SOA | 管理这个域名的七个重要参数 |

| | | | |
|-------|----|-------|-------------------|
| 域名. | IN | MX | 顺序数字, 接收邮件的服务器主机名 |
| 主机别名. | IN | CNAME | 实际代表这个主机别名的主机名 |

2、**A、AAAA(Address)**: 查询 IP 的记录

- dig -t a www.ksu.edu.tw
- 这个 A 的 RR 类型是在查询某个主机名的 IP, 也是最常被查询的一个 RR 标志
- **如果主机是主机名, 结尾部分务必加上小数点???**
- 如果 IP 设置的是 IPv6 的话, 那么查询就需要使用 AAAA 类型才行

3、**NS(Name Server)**: 查询管理区域名(Zone)的服务器主机名

- dig -t ns ksu.edu.cn

4、**SOA(Start of Authority)**: 查询管理域名的服务器管理信息

- 如果有多台 DNS 服务器管理同一个域名, 那么最好使用 Master/Slave 的方式来进行管理, 就需要声明被管理的 zone file 是如何进行传输的, 此时就需要 SOA(Start Of Authority)的标志了

➤ dig -t soa ksu.edu.tw

; ; ANSWER SECTION:

```
ksu.edu.tw.    3600    IN  SOA dns1.ksu.edu.tw.      abuse.mail.ksu.edu.tw.
2014010157 1800 900 604800 86400
```

- SOA 后面总会接七个参数

- 1) Master DNS 服务器主机名: 这个区域主要是哪台 DNS 作为 Master 的意思
- 2) 管理员的 E-mail: 由于@ 在数据库文件中是有特殊意义的, 因此就将@ 替换为 mail
- 3) 序号(Serial): 当修改了数据库内容时, 需要将这个值放大才行, 通常的格式为"YYYYMMDDNU", 例如 2010080369 代表 2010 年 08 月 03 当天第 69 次更改
- 4) 更新频率(Refresh): Slave 向 Master 要求数据更新的频率, 如果发现序号没有增大, 则不会下载数据库文件
- 5) 失败重新尝试时间(Retry): 由于某些因素导致 Slave 无法对 Master 实现连接, 那么在多长时间内, Slave 会尝试重新连接到 Master, 若连接成功了, 又恢复为原来的更新频率
- 6) 时效时间(Expire): 如果一直尝试失败, 持续连接到这个设置值时限, 那么 Slave 将不再继续尝试连接, 并尝试删除这份下载的 zone file 信息
- 7) 缓存时间(Mimumum TTL): 如果数据库 zone file 中, 每笔 RR 记录都没有写到 TTL 缓存时间的话, 那么就以 SOA 的设置值为主

- 参数限制

- Refresh>=Retry*2
- Refresh+Retry<Expire
- Expire>=Retry*10
- Expire>=7Days

5、**CNAME(Canonical Name)**: 设置某主机名的别名(alias)

- 有时候你不想针对某个主机名设置 A 标志, 而是想通过另外一台主机名的 A 来规范这个新主机名, 这时可以使用 CNAME 的设置
- 为什么要用 CNAME?

- 如果你有一个 IP，这个 IP 是给很多主机名使用的，那么当你的 IP 更改时，所有的数据就得全部更新 A 标志才行
- 如果你只有一个主要主机名设置 A，而其他的标志使用 CNAME，那么当 IP 更改时你只需要修订一个 A 标志，其他的 CNAME 就跟着变动了，处理起来比较容易

6、**MX(Mail Exchanger)**: 查询某域名的邮件服务器主机名

- MX 是 Mail eXchanger(邮件交换)的意思
- 通常整个区域会设置一个 MX，表示所有寄给这个区域的 E-mail 应该要送到后头的 E-mail Server 主机名上才行
- 通常大型的企业会有多台上层的邮件服务器来预先接受信件，那么数字较小的邮件服务器优先

19.4.2. 反解文件记录的 RR 数据

1、以 www.ksu.edu.tw.为例，从整个域的概念来看，越右边出现的名称代表域越大，即.(root)>.tw>.edu

2、但是 IP 不同，以 120.114.100.101 为例，120>114>100>101，与默认的 DNS 从右边向左边查询不一样，**于是反解的 Zone 必须要将 IP 反过来，而在结尾时加上.in-addr-arpa.的结尾字样即可**

```
> dig -x 120.114.100.101 <==得到以下输出
;; ANSWER SECTION:
101.100.114.120.in-addr.arpa. 3600 IN PTR www.ksu.edu.tw.
```

3、PRT 就是反解，即是查询 IP 所对应的主机名

➤ **反解最重要的地方就是：后面的主机名尽量使用完整的 FQDN，亦即机上小数点(.)，**

19.4.3. 步骤一：DNS 的环境规划

1、假设在局域网中要设置 DNS 服务器，局域网域名为 centos.liuye，搭配的 IP 网段为 192.168.100.0/24，因此主要正解区域为 centos.liuye，反解的区域是 192.168.100.0/24。并且想要寻找(.)而且回通过 forwarders 的辅助，因此还需要".."的区域正解文件，以下为需要的配置文件

- named.conf(主要配置文件)
- named.centos.vbird(主要的 centos.vbird 的正解文件)
- named.192.168.100(主要的 192.168.100.0/24 的反解文件)
- named.ca(由 bind 软件提供的.正解文件)

19.4.4. 步骤二：主配置文件/etc/named.conf 的设置

1、该文件的主要任务

- options: 规范 DNS 服务器的权限(可否查询、forward 与否等)
- zone : 设置出 Zone(domain name) 以及 zone file 的所在 (包含 master/slave/hint)
- 其他: 设置 DNS 本机管理接口以及相关的密钥文件(key file)

2、以下为配置内容

```
options {
    listen-on port 53 { any; };
```

```

directory      "/var/named";
dump-file     "/var/named/data/cache_dump.db";
statistics-file "/var/named/data/named_stats.txt";
memstatistics-file "/var/named/data/named_mem_stats.txt";
allow-query      { any; };

recursion yes;

allow-transfer {none;};
};

zone "." IN{
    type hint;
    file "named.ca";
};

zone "centos.liuye." IN{
    type master;
    file "named.centos.liuye";
};

zone "200.168.192.in-addr.arpa." IN {
    type master;
    file "named.192.168.200";
};

```

19.4.5. 步骤三：最上层. (root) 数据库文件的设置

- 1、"."是由 INTERNIC 所管理维护的，全世界共有 13 台管理"."的 DNS 服务器
- 2、BIND 软件已经提供了一个名为 named.ca 的文件了

19.4.6. 步骤四：正解数据库文件的设置

- 1、正解文件一定要有的 RR 标志有以下几个

- 关于本区域的基础设置方面：例如缓存记忆时间(TTL)、域名(ORIGIN)等
- 关于 Master/Slave 的认证方面(SOA)
- 关于本区域的域名服务器所在的主机名与 IP 对应(NS、A)
- 其他正反解相关的资源记录(A、MX、CNAME 等)

- 2、正解文件中的特殊符号

| 字符 | 意义 |
|---------|--|
| 一定从行首开始 | 所有设置数据一定要从行首开始，前面不可有空格符，若有空格符，代表延续前一个 domain 的意思，非常重要 |
| @ | 代表 Zone 的意思，例如写在 named.centos.liuye 中，@代表 centos.liuye. (注意最后的点)，如果写在 named.192.168.200 文件中，则@代表 200.168.192.in-addr.arpa.(注意最后的点) |
| . | 非常重要，因为它带包一个完整主机名(FQDN)而不是仅有 |

| | |
|---|---|
| | hostname 而已 例如在 named.centos.liuye 当中写 www.centos.vbird 则 FQDN 为 www.centos.liuye.@==>www.centos.liuye.centos.liuye.，因此自然要写成 www.centos.liuye. |
| ; | 代表批注符号，似乎#也是批注，两个都能用 |

3、以下为配置内容

```
$TTL 600
@ IN SOA master.centos.liuye. liuye.www.centos.liuye.
(2011080401 3H 15M 1W 1D)
```

```
@ IN NS master.centos.liuye.
master.centos.liuye. IN A 192.168.200.254
@ IN MX 10 www.centos.liuye.
```

```
www.centos.liuye. IN A 192.168.200.254
linux.centos.liuye. IN CNAME www.centos.liuye.
ftp.centos.liuye. IN CNAME www.centos.liuye.
```

```
Linux2.centos.liuye. IN A 192.168.200.101
```

4、务必注意完整主机名(FQDN)之后的点

- 加上了"."表示这是个完整的主机名(FQDN)，亦即 hostname+domain name
- 若没有加上".", 表示该名称仅为 hostname, 因此完整的 FQDN 要加上 Zone, 即 **hostname.@**

19.4.7. 步骤五：反解数据库文件的设置

1、由于反解的 Zone 名称是"zz.yy.xx.in-addr.arpa."模样，因此只要在反解里面用到主机名时，务必使用 FQDN 来设置

2、由于我们的 Zone 是"200.168.192.in-addr.arpa."，因此 IP 的全名部分已经含有 192.168.200 了，因此我们只需要写出最后一个 IP 即可，而且不用加点，利用 **hostname.@** 来帮助我们补全

3、以下为配置内容

```
$TTL 600
@ IN SOA master.centos.liuye. liuye.www.centos.liuye.
(2011080401 3H 15M 1W 1D)
@ IN NS master.centos.liuye.
254 IN PTR master.centos.liuye.
```

```
254 IN PTR www.centos.liuye.
101 IN PTR Linux2.centos.liuye.
```

- 同理反解的主机名也得记得自后加".", 否则给你自动补上".@"了

19.4.8. 步骤六：DNS 启动查看与防火墙

1、systemctl restart named.service

- 2、`systemctl enable named.service`
- 3、利用 `firewall-cmd` 或者 `firewall-config` 来配置防火墙

19.4.9. 测试与数据库更新

- 1、测试有两种方式
 - 一种通过 Client 端的查询功能，目的是检验数据库设置有无错误
 - 利用 <http://thednsreport.com/> 来检验，但是该网站的检验主要是以合法授权的 Zone 为主，我们自己乱搞的 DNS 是没法检验的
- 2、修改 `/etc/resolv.conf`
 - 使得自己的非合法 DNS 主机 IP 写在最上面，否则只要上面的 DNS 网络通畅，那么该 DNS 便是无效的，因此自然查不到自己设定的非法主机名
- 3、判断成功与否
 - 是否与预期相符合
 - 如果出现错误信息或者找不到某个反解的 IP 或者正解的主机名那么就说明不正确了
- 4、**数据库更新，例如主机名或 IP 变更，或者添加某个主机名与 IP 对应**
 - 先针对要更改的那个 Zone 的数据库文件去做更新，就是加入 RR(Resource Record)标志
 - 更改该 zone file 的序号(Serial)，就是 SOA 的第三个参数(第一个数字)，因为这个数字会影响到 Master/Slave 的判定更新与否
 - 重新启动 named，或者让 named 重新读取配置文件即可

19.5. 协同工作的 DNS：Slave DNS 及子域授权设定

- 1、Slave DNS 的特色
 - 为了不间断地提供 DNS 服务，你的领域至少需要有两台 DNS 服务器来提供查询的功能
 - 这几台 DNS 服务器应该要分散在两个以上的不同 IP 网段才好
 - 为方便管理，通常除了一台主要 Master DNS 之外，其他的 DNS 会使用 Slave 模式
 - Slave DNS 服务器本身并没有数据库，它的数据库是由 Master DNS 所提供的
 - Master/Slave DNS 需要有可以相互传输 zone file 的相关信息才行，这部分需要 `/etc/named.conf` 的设置加以辅助

19.5.1. masterDNS 权限的开放

- 1、基本假设
 - 提供 Slave DNS 服务器进行 zone transfer 的服务器为 `master.centos.liuye`
 - `centos.liuye` 以及 `200.168.192.in-addr.arpa` 两个 Zone 都提供给 Slave DNS 使用
 - `master.centos.vbird` 的 named 仅提供给 `slave.centos.liuye` 这台主机进行 zone transfer
 - Slave DNS Server 架设在 `192.168.200.101` 这台服务器上面(所以 zone file 要修订)

2、将上面的 Master 主机修改后如下

```
options {
    listen-on port 53 { any; };
    directory    "/var/named";
    dump-file    "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    allow-query     { any; };

    recursion yes;

    allow-transfer {none;};
};

zone "." IN{
    type hint;
    file "named.ca";
};

zone "centos.liuye." IN{
    type master;
    file "named.centos.liuye";
    allow-transfer { 192.168.200.101; };
};

zone "200.168.192.in-addr.arpa." IN {
    type master;
    file "named.192.168.200";
    allow-transfer { 192.168.200.101; };
};
```

19.5.2. Slave DNS 的设置与数据库权限问题

- 既然 Slave DNS 也是 DNS 服务器，所以也要安装 bind、bind-chroot 等软件
- 既然 Master/Slave 的数据库是相同的，那么理论上/etc/named.conf 内容就是大同小异了

- 唯一要注意的就是 zone type 类型，以及声明 Master 在哪里
- 至于 zone filename 部分，由于 zone file 都是从 Master 取得的，通过 named 这个程序来主动建立起需要的 zone file，因此这个 zone file 放置的目录权限就很重要

- 以下是/etc/named.conf 配置内容

```
options {
    listen-on port 53 { any; };
    directory    "/var/named";
```

```

dump-file    "/var/named/data/cache_dump.db";
statistics-file "/var/named/data/named_stats.txt";
memstatistics-file "/var/named/data/named_mem_stats.txt";
allow-query      { any; };

recursion yes;

allow-transfer {none;};
};

zone "." IN{
    type hint;
    file "named.ca";
};

zone "centos.liuye." IN{
    type slave;
    file "slaves/named.centos.liuye";
    masters { 192.168.200.254; }; //注意大括号与 IP 之间有空格，不然会高亮
    红色显示
};

zone "200.168.192.in-addr.arpa." IN {
    type slave;
    file "slaves/named.192.168.200";
    masters { 192.168.200.254; };
}

```

19.5.3. 配置子域 DNS 服务器：子域授权课题

1、我们以刚在 Master 上面建立的 centos.liuye 这个 Zone 为例，假设今天你是个 ISP，有人想和你申请 domain name，他要的 domain 是 "lh.centos.liuye"，该如何处理

- 上层 DNS 服务器：也就是 master.centos.liuye 这一台，只要在 centos.liuye 那个 zone file 内，增加指定 NS 并指向子层 DNS 主机名与 IP 对应即可，而 zone file 的需要也要增加才行
- 下层 DNS 服务器：申请的域名必须是上层 DNS 所可以提供的名称，并告知上层 DNS 管理员，我们这个 Zone 所需指定的 DNS 主机名与对应的 IP 即可，然后就可以开始设置自己的 Zone 与 zone file 相关数据了

2、上层 DNS 服务器：只需要添加 zone file 的 NS 与 A 即可

- /etc/named.conf 中添加如下两行
- ```

lh.centos.liuye. IN NS dns.lh.centos.liuye.
dns.lh.centos.liuye. IN A192.168.200.102

```

3、下层 DNS 服务器：需要有完整的 Zone 相关设置

- /etc/named.conf 中内容如下

```

options {
 listen-on port 53 { any; };
 directory "/var/named";
 dump-file "/var/named/data/cache_dump.db";
 statistics-file "/var/named/data/named_stats.txt";
 memstatistics-file "/var/named/data/named_mem_stats.txt";
 allow-query { any; };

 recursion yes;

 allow-transfer {none;};
};

zone "." IN{
 type hint;
 file "named.ca";
};

zone "lh.centos.liuye." IN{
 type master;
 file "named.lh.centos.liuye";
};

```

- /var/named/named.lh.centos.liuye 内容如下

```

$TTL 600
@ IN SOA dns.lh.centos.liuye. root.lh.centos.liuye.
 (2016112901 3H 15M 1W 1D)
@ IN NS dns.lh.centos.liuye.
dns IN A192.168.200.101

```

#### 19.5.4. 依不同接口给予不同的 DNS 主机名：view 功能的应用

1、以目前的局域网服务器来说，我的 master.centos.liuye 有两个接口，分别是 192.168.200.254/24(对内)以及 192.168.136.166(对外)

- 外边的用户想要了解到 master.centos.liuye 这台服务器的 IP 时，取得的是 192.168.200.254，因此需要通过 NAT 才能连接到该接口
- 但是明明 192.168.136.166 与 192.168.200.254 是同一台服务器主机，有没有办法让外部查询找到 master.centos.vbird 是 192.168.136.166 而内部的找到则回应 192.168.200.254---通过 view 功能

2、设置 view

- 建立一个名为 intranet 的名字，这个名字代表客户端为 192.168.200.0/24 的来源
- 建立一个名为 internet 的名字，这个名字代表客户端为非 192.168.136.0/24 的其他来源
- intranet 使用的 zone file 为前面所建立的 zone filename，internet 使用的

zone filename 则在原本的文件名后面累加 inter 的扩展名，并修订各标志的结果

- 最终的结果中，从内网查到的 www.centos.liuye IP 应该是 192.168.200.254，从外网中查询到的 www.centos.liuye IP 应该是 192.168.136.166

### 3、以下为修改后的/etc/named.conf

```
options {
 listen-on port 53 { any; };
 directory "/var/named";
 dump-file "/var/named/data/cache_dump.db";
 statistics-file "/var/named/data/named_stats.txt";
 memstatistics-file "/var/named/data/named_mem_stats.txt";
 allow-query { any; };

 recursion yes;

 allow-transfer {none;};
};

acl intranet { 192.168.200.0/24; };
acl internet { ! 192.168.200.0/24; any; };

view "lan" {
 match-clients { "intranet"; };
 zone "." IN{
 type hint;
 file "named.ca";
 };

 zone "centos.liuye." IN{
 type master;
 file "named.centos.liuye";
 allow-transfer { 192.168.200.101; };
 };

 zone "200.168.192.in-addr.arpa." IN {
 type master;
 file "named.192.168.200";
 allow-transfer { 192.168.200.101; };
 };
}

view "wan" {
 match-clients { "internet"; };
```

```

zone "." IN {
 type hint;
 file "named.ca";
};

zone "centos.liuye." IN{
 type master;
 file "named.centos.liuye.inter";
};
};

```

## 19.6. DNS 服务器的高级设定

### 19.6.1. 架设一个合法授权的 DNS 服务器

- 1、必须要上层服务器将子域的查询权开放给你来设置
- 2、申请一个合法的 domain name(缴费)
- 3、以 DNS 服务器的详细设置的内容来设置主机
- 4、测试

### 19.6.2. LAME Server 的问题

- 1、当 DNS 服务器在向外面的 DNS 系统查询某些正反解时，可能由于对方 DNS 主机的设置错误，导致无法解析到预期的正反解结果，这时候就是所谓的 Lame Server 的错误

### 19.6.3. 利用 RNDC 命令管理 DNS 服务器

- 1、rndc 是 BIND version9 以后所提供的功能，可以让你轻松管理 DNS 服务器，包括检查已经存在 DNS 缓存当中的资料，重新更新某个 Zone 而不需要重新启动整个 DNS，以及检查 DNS 的状态与统计资料等
- 2、由于 rndc 可以很深入地管理你的 DNS 服务器，所以要进行一些控制
  - 通过 rndc 的设置来建立密钥(rndc key)，并将密钥相关的信息写入 named.conf 配置文件当中
  - 通过以下步骤进行配置(详见 P639-P640)
    - rndc-confgen
    - 然后将输出的信息中的 key "rndc-key" {...};部分复制
    - vim /etc/rndc.key
    - 将复制的信息覆盖掉原内容
    - vim /etc/named.conf
    - 在最后面添加上述复制的内容
    - 然后追加以下
 

```
controls{
 inet 127.0.0.1 port 953 allow { 127.0.0.1; } keys { "rndc-key"; };
}
```

#### 19.6.4. 搭建动态 DNS 服务器：让你成为 ISP

1、如果以拨号方式的 ADSL 连上 Internet，那么 IP 通常是 ISP 随机提供的，因此，每次上网的 IP 都不固定，因此 DDNS(Dynamic DNS,DDNS)主机就必须提供一个机制，让客户端可以通过这个机制来修改他们在 DDNS 主机上面的 zone file 内的数据才行

#### 2、如何实现

- BIND9 就提供类似的机制
- 我们的 DDNS 主机现提供 Client 一个 key(就是认证用的数据，你可以将它理解成账号与密码的概念)
- Client 端利用这个 key，并配合 BIND9 的 nsupdate 命令，就可以连上 DDNS 主机，并且修改主机上面 zone file 内的对应表了

#### 3、DDNS Server 端的设置

- <dnssec-keygen>
  - dnssec-keygen -a [算法] -b [密码长度] -n [类型] 名称
  - -a: 后面接的 type 为盐酸方式，主要有 RSAMD5、RSA、DSA、DH、HMAC-MD5 等
  - -b: 密码长度，通常给予 512 位的 HMAC-MD5
  - -n: 后面接的则是客户端能够更新的类型，主要有下面两种
    - ZONE: 客户端可以更新任何标志及整个 ZONE
    - HOST: 客户端仅可以针对他的主机名来更新
- cd /etc/named
- dnssec-keygen -a HMAC-MD5 -b 512 -n HOST web
- 接下来将公钥的密码复制到 /etc/named.conf 中，将私钥传给 web.centos.liuye 那台主机
- vim /etc/named.conf

```
key "web" {
 algorithm hmac-md5;
 secret "<这里是公钥密码>";
};

zone "centos.liuye." IN{
 type master;
 file "named.centos.liuye";
 allow-transfer { 192.168.200.101; };
 update-policy{ #添加这一段
 grant web name web.centos.liuye. A;
 };
};
```
- grant [key\_name] name [hostname] 标签 <==允许指定主机通过指定密码修改指定标签
- chmod g+w /var/named
- chown named /var/named/named.centos.liuye
- systemctl restart named.service
- setsebool -P named\_write\_master\_zones=1

#### 4、设置 Client 端

- 将 Server 端的密码公钥与私钥文件通过 sftp 传送到客户端的 /usr/local/ddns 目录下
- cd /usr/local/ddns
- nsupdate -k \*.key <==跟公钥文件名
  - server 192.168.200.254
  - update add web.centos.liuye 600 A 192.168.200.102
  - send
  - [ctrl]+D <==退出
- 于是就会发现在 DNS 服务器端的/var/named/里面多出一个临时文件，那就是 named.centos.liuye.jnl，用于记录客户端要求而更新的数据

## Chapter 20. WWW 服务器

### 20.1. WWW 的简史、资源以及服务器软件

#### 20.1.1. WWW 的简史、HTML 与标准制定 (W3C)

1、WWW 是 World Wide Web 的缩写，其中 Web 有广播网的意思，所以简称为全球信息网，WWW 可以结合文字、图形、影像以及声音等多媒体，并通过鼠标单击超链接(Hyperlink)的方式将信息以 Internet 传递到世界各处

2、与其他服务器类似，当你连上 WWW 网站时，该网站肯定会提供一些数据，而你的客户端必须要使用可解析这些数据的软件来处理，那就是浏览器

#### 3、概念

- WWW 服务器不但需要可以让客户端浏览的平台，还需要提供客户端一些数据才行
- 服务器所提供的最主要数据时超文本标记语言 (Hyper Text Markup Language, HTML)、多媒体文件(图片、影像、声音、文字等，都属于多媒体或称为超媒体)
- HTML 只是一些纯文本数据，通过所谓的标记<tag>来规范所要显示的数据格式
- 在客户端，浏览器通过对 HTML 以及多媒体数据进行解析，最后将效果呈现在用户的屏幕上

#### 4、HTML 的格式

- 服务器提供客户端的一些数据，主要都以 HTML 格式来呈现
- HTML 由许多<tag>组成

#### 5、WWW 所用的协议及 WWW 服务器简史

- 超文本传输协议(Hyper Text Transport Protocol,HTTP)

#### 6、浏览器(browser)大战与支持的标准

- 20 世纪 90 年代末微软将 IE 浏览器内建在 Windows 操作系统内
- FireFox 发展就标榜小而美，因此程序相当小，执行效率上非常快速，FireFox 主要是依据 W3C 所指定的标准来发展的

#### 7、总结

- WWW 是依据 HTTP 这个协议而来的，分为服务器端与客户端
- Apache 是一个服务器端的软件，主要依据 NCSA 的 HTTPd 服务器发展而来，为自由软件
- Mozilla 是一个自由软件的开发计划，其中 FireFox 浏览器是相当成功的作品
- 在撰写自己的网页数据时，尽量使用 W3C 所发布的标准，这样在所有浏览器上面才能够顺利的显示出你想要的样子

#### 20.1.2. WWW 服务器与浏览器所提供的资源定位 (URL)

1、我们需要在服务器端将数据文件写好，并且放置在某个特殊的目录下面

- 这个目录就是我们整个网站的首页了
- 这个目录很可能在/var/www/html/或者/srv/www/
- CentOS 默认在/var/www/html/

2、浏览器如何取得目录内的数据

- 必须要在浏览器的地址栏中输入所需要的网址才行，这个网址就对应到 WWW 服务器的某个文件的文件名
  - 现在浏览器功能很多，不止可以连上 WWW，还可以连上 FTP 之类的网络协议，所以需要在地址栏输入正确的网址

<协议>://<主机名或主机地址>[:port]/<目录资源>

### 3、网址的意义

- URL(Uniform Resource Locator), 以斜线作为分段, 分为以下几个部分
  - 协议
    - 浏览器比较常支持的协议有 HTTP、HTTPs、FTP、Telnet 等
  - 主机地址或主机名
    - 就是服务器在因特网所在的 IP 位置, 如果是主机名, 还需要通过名称解析器
    - 一般来说, 使用 IP 就能假设 WWW 网站, 不过建议还是申请一个好记又合法的主机名
  - 目录资源
    - 举例来说, 若 WWW 数据放置在 /var/www/html/ 中
    - http://linux.vbird.org --> /var/www/html/
    - http://linux.vbird.org/linux\_basic/index.php  
                                -> /var/www/html/linux\_basic/index.php
    - 另外, 通常首页目录下面会有一个特殊的文件名, 例如 index.html 或 index.??? 等 , 即 输入 http://linux.vbird.org 与 输入 http://linux.vbird.org/index.php 是一样的

#### 4、WWW Server/Client 间数据传输的方式

- 浏览器与服务器端传递数据的方式有以下几种
  - GET
    - 浏览器直接向 WWW 服务器要求网址上面的资源，这是最常见的
    - 另外使用 GET 的方式可以直接在网址列输入变量，例如 <http://phorum.vibrd.org/viewtopic.php?t=96>, t 就是变量，96 就是变量内容
  - POST
    - 这也是客户端向服务器端提出的要求，只是这个要求里面含有比较多的数据
    - POST 与 GET 不同，**GET 可以在网址取得客户端所要求的的变量**，POST 就不是利用网址的功能了，相当于隐藏了具体请求信息
  - HEAD
    - 服务器端响应给 Client 端的一些数据头文件而已
  - OPTIONS
    - 服务器端响应给 Client 端的一些允许的功能与方法
  - DELETE
    - 删除某些资源的举动

### 20.1.3. WWW 服务器的类型：系统、平台、数据库与程序 (LAMP)

1、目前来说，市场占有率较高的 WWW 服务器软件应该是 Apache 与 IIS 这两个，Apache 是自由软件，可以在任何操作系统上面安装，IIS 则是 Windows 开发出来，

仅能在 Windows 操作系统上面安装与执行

## 2、仅提供用户浏览的单向静态网页

- 这种类型的网站大多是提供"单向静态"的网页，或许还提供一些动画显示，但基本上仅止于此
- 因为单纯是由服务器单向提供数据给客户端，Server 不需要与 Client 端有互动，所以你可以到该网站上去浏览，但是无法进行数据的上传
- 目前主要的免费虚拟主机大多是这样的类型，只要依照 HTML 的语法写好网页，并且上传到该网站空间上，那么你的数据就能够让大家浏览了

## 3、提供用户互动接口的动态网站

- 这种类型的网站可以让服务器与用户互动，常见的例如讨论区与留言板，包括一些博客也属于这类型
- 这种类型的网站需要的技术程度比较高，因为它是通过"网页程序语言"来实现与用户互动的行为
- 常见的例如 PHP 网页程序语言，配合 MySQL 数据库系统来进行数据的读写
- 不论你要求的数据是什么，其实都是通过服务器端同一个网页程序在负责将数据库读取或写入数据库，处理完毕后将结果传给客户端的一种方式，变动的是数据库内的数据，网页程序其实没有任何改变，这部分网页程序包括 PHP、ASP、Perl 等

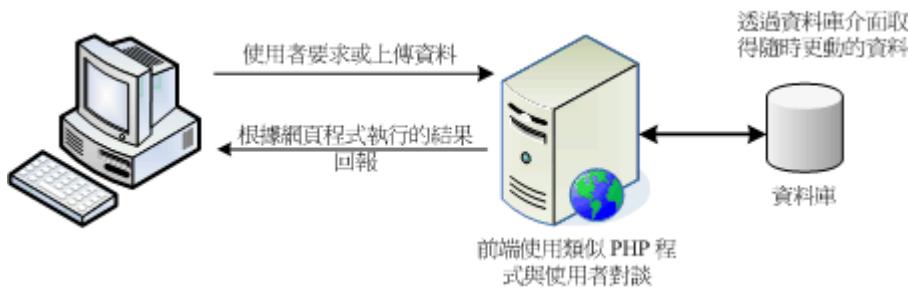


图 20-1 动态网站的网页程序语言与数据库接口

### ➤ 另一种交互式的动态网页主要是在客户端实现的

- 我们可以通过利用所谓的 JavaScript 这种语法，将可执行的程序代码 (JavaScript) 传送给客户端，客户端的浏览器如果提供 JavaScript 的功能，该程序就可以在客户端的计算机上面工作了
- 由于程序时在客户端计算机上面执行，因此如果服务器端所制作的程序是恶意的，那么客户端的计算机就可能会遭到破坏
- 另外一种可在客户端执行的就是 Flash 动画格式，在这种动画格式内还可以进行程序设计，因此客户端只要拥有可执行 Flash 动画的软件，就可以利用这个软件来实现与用户的交互
- 动态网站需要具备的条件
  - 支持的操作系统：让所需要的软件都能够安装执行
  - 可运行的 WWW 服务器，例如 Apache 与 IIS 等 WWW 服务器平台软件
  - 网页程序语言：包括 Perl、PHP、JSP、CGI、ASP 等
  - 数据存储的数据库系统：包括 MySQL、MSSQL、PostgreSQL 以及甲骨文 (Oracle) 等

## 4、LAMP 平台的说明

- 在整个平台设计上面，目前常见的有两大系统

- 一个是 Linux 操作系统上面，搭配 Apache+MySQL+PHP 等而实现，这个系统被称为 LAMP
- 另一个则是微软的 IIS+MSSQL+ASP(.NET)服务器
- 在能见度与市场占有率方面，应该还是 LAMP 为主
- Apache
  - 1995 年之前，WWW 服务器软件以 HTTPd 占有率较高
  - 后来 HTTPd 经过多次修订后，在 1995 年发布了 Apache(A patch server)
  - PHP 必须要在这上面才能运行
- MySQL
  - 数据库是一种特殊格式的文件，这种文件需要通过特殊接口(数据库软件)来进行读写
  - 这个特殊接口已经针对数据的查询、写入做过优化设计，因此适合多人同时写入与查询的工作
  - 数据库的语法有所谓的 SQL 标准语法，任何根据这种数据检索语法发展出来的数据库，都称为 SQL 数据库，比较知名的自由软件数据库系统有 MySQL 以及 PostgreSQL，其中 MySQL 使用率又较高一些
- PHP
  - PHP 可以被用来建立动态网页，PHP 程序代码可以直接在 HTML 网页中嵌入，就像编辑 HTML 网页一样简单
  - 这种程序语言可以直接在网页中编写，不需要经过编译即可进行程序的执行，由于具有自由软件、跨平台、容易学习以及执行效率高等有点，目前是很热门的一个设计网页的工具

#### 20.1.4. https：加密的网页数据(SSL)及第三方证书机构

1、**HTTP 这个传输协议是以明码传送的**，所以你的任何数据包只要被监听窃取的话，那么该数据等于是别人的

2、一些关于你个人重要机密的数据就不能这样随意传送，这时候就需要用到 <https://hostname> 这种连接方式了，这种方式采用的是 SSL 加密机制

##### 3、Secure Socket Layer(SSL)

- 当浏览器与 WWW 服务器端同时支持 SSL 的传输协议时，在连接阶段浏览器与服务器就会产生那把重要的密钥
- 产生密钥后就能够利用浏览器来传送与接收加密过的重要数据

##### 4、Certificate Authorities(CA)

- SSL 机制的问题就是：那把 Public Key 是服务器产生且任何人都能取得的
- 所谓 CA 就是一个公认的公证单位，你可以自行产生一把密钥且制作出必要的证书数据并向 CA 单位注册(要钱的)，那么当客户端浏览器在浏览时，该浏览器会主动向 CA 单位确认该证书是否为合法注册过的，如果是的话，那么该次连接才会建立

#### 20.1.5. 客户端常见的浏览器

1、由于浏览器可以接连到因特网，所以浏览器也有可能被攻击，由于 IE 内子安在 Windows 内核当中，所以如果 IE 有漏洞，对于系统的损害是很大的

2、除了窗口接口的浏览器软件之外，还有如下几个可以在文字接口进行浏览与网页下载的程序

- links 与 lynx: 文字接口浏览器
- wget: 文字接口用来下载文件的命令

## 20.2. WWW(LAMP) 服务器基本配置

### 20.2.1. LAMP 所需软件与其结构

1、由于我们是 CentOS，那么只需要 CentOS 本身提供的 Apache、PHP、MySQL 即可，**不建议自行利用 tarball 安装 LAMP 服务器**，除非你有特殊需求(某些 Apache 需要较高版本，或者 PHP、MySQL 有特殊版本的需求)

- yum install httpd mariadb mariadb-server php php-mysql
- 其中 mariadb 和 mariadb-server 是 CentOS7 下的 SQL 软件
- 由于 CentOS7 上没有 mysql 的源，因此增加 mysql 源
  - wget http://repo.mysql.com/mysql-community-release-el7-5.noarch.rpm
  - rpm -ivh mysql-community-release-el7-5.noarch.rpm
- yum install mysql mysql-server
- **但是，自从添加该源后，如果将其卸载，重新安装 mariadb，则 mariadb 无法启动，不知道什么原因!!!**

2、PHP 是挂在 Apache 下面执行的一个模块，而我们要用网页的 PHP 程序控制 MySQL 时，PHP 就需要支持 MySQL 的模块才行，至少需要以下软件

- httpd(提供 Apache 主程序)
- mysql(MySQL 客户端程序)
- mysql-server(MySQL 服务器程序)
- php(PHP 主程序含给 Apache 使用的模块)
- php-devel(PHP 的发展工具，这个与 PHP 外挂的加载速度软件有关)
- php-mysql(提供给 PHP 程序读取 MySQL 数据库的模块)

3、Apache 软件的相关结构

- /etc/httpd/conf/httpd.conf: 主要配置文件
- /etc/httpd/conf.d/\*.conf: 很多的额外参数文件，扩展名是.conf
- /usr/lib64/httpd/modules/、/etc/httpd/modules/: 外挂模块放置目录
- /var/www/html/: 输入 http://localhost 时显示的默认数据就是放在这个目录中的首页文件(默认为 index.html)
- /var/www/error/: 错误信息的目录
- /var/www/icons/: 提供 Apache 默认给予的一些小图示，输入 http://localhost/icons/ 时所显示的数据所在
- /var/www/cgi-bin/: 默认给一些可执行 CGI 程序放置的目录，输入 http://localhost/cgi-bin/ 时所显示的数据所在
- /var/log/httpd/: 默认的 Apache 日志文件都在这里
- /usr/sbin/apachectl: Apache 主要的执行文件，就是一个 Shell Script
- /usr/sbin/httpd: 主要的 Apache 二进制执行文件
- /usr/bin/htpasswd: Apache 密码保护

4、MySQL 软件相关结构

- /etc/my.cnf: MySQL 的配置文件，包括你想要进行 MySQL 数据库的优化，或者是针对 MySQL 进行一些额外的参数指定
- /var/lib/mysql/: 这个目录则是 MySQL 数据库文件存储的所在

## 5、PHP 软件相关结构

- /etc/httpd/conf.d/php.conf:
- /etc/php.ini: PHP 的主要配置文件，包括 PHP 能不能允许用户上传文件，能不能允许某些低安全性的标志等
- /usr/lib64/httpd/modules/libphp5.so: PHP 提供给 Apache 使用的模块，这也是能否在 Apache 网页上面设计 PHP 程序语言的最重要文件
- /etc/php.d/mysql.in、/usr/lib64/php/modules/mysql.so: PHP 支持 MySQL 接口
- /usr/bin/phpize、/usr/include/php/

### 20.2.2. Apache 的基本设定

1、如果是暂时测试试用的主机而没有主机名时，那么至少确定测试用主机名为 localhost 且在/etc/hosts 内需要有如下内容

- 127.0.0.1 localhost.localdomain localhost

2、**/etc/httpd/conf/httpd.conf** 配置文件的基本格式如下

<设置项目>

此设置项目内的相关参数

.....

</设置项目>

3、针对服务器环境的设置项目

- ServerRoot "/etc/httpd": 服务器设置的最顶层目录，包括 logs、modules 等数据都需要放在此目录下面
- Listen 80: 与监听的接口有关，默认开放在所有的网络接口，也可以修改端口
- User apache: 属主
- Group apache: 属组
- ServerAdmin root@localhost: 系统管理员的邮箱

4、针对中文 Big 语言编码的设置参数修改

- 目前的因特网多以(utf-8)为主
- 若要以浏览器指定的编码来正确显示，那么只需要注释掉 AddDefaultCharset 设置项即可

5、网页首页以及目录相关权限的设置(DocumentRoot 与 Directory)

- 由于 Apache 允许 Internet 对我们的数据进行浏览，所以必须要针对可被浏览的目录进行权限的相关设置，这就是<Directory>设置值
- DocumentRoot "/var/www/html" <==可以改成你放置首页的目录
  - 这个设置规范了 WWW 服务器主网页所放置的目录，虽然设置值内容可以变更，但是必须要留意这个目录的权限以及 SELinux 的相关规则与类型
- <Directory>  
<Directory "/var/www/html">  
    Options Indexes FollowSymLinks  
    AllowOverride None  
    Order allow,deny  
    Allow from all

</Directory>

- Options(目录参数): 此设置值表示在这个目录内能够让 Apache 进行的操作，也就是针对 Apache 程序的权限设置，主要参数如下
    - Indexes: 如果在此目录下找不到首页文件(默认为 index.html)时，就显示整个目录下的文件名，**至于首页文件名泽字 DirectoryIndex 设置值有关**
    - FollowSymLinks: 让连接文件可以生效，由于 /var/www/html 是 WWW 的根目录，因此一般来说，被 chroot 的程序无法离开该目录，但是此设置值可以让连接文件有效地离开本目录
    - ExecCGI: 让此目录具有执行 CGI 程序的权限，非常重要
    - Includes: 让一些 Server-Side Include 程序可以运行
    - MultiViews: 与语言数据有关
  - AllowOverride(允许的覆盖参数功能): 表示是否允许额外配置文件 .htaccess 的某些参数覆盖，Apache 默认可以让用户以目录下面的 .htaccess 文件内覆盖 <Directory> 内的某些功能参数，该项目是在规定 .htaccess 可以覆盖的权限类型有哪些
    - ALL: 全部的权限均可被覆盖
    - AuthConfig: 仅有网页认证(账号与密码)可覆盖
    - Indexes: 仅允许 Indexes 方面的覆盖
    - Limits: 允许用户利用 Allow、Deny 与 Order 管理可浏览的权限
    - None: 不可覆盖，让 .htaccess 文件失效
  - Order、allow、deny(能否登录浏览的权限): 决定此目录是否可悲 Apache 的 PID 所浏览的权限设置，能否被浏览主要有两种判断方式
    - deny, allow: 以 deny 优先处理，但没有写入规则的则默认为 allow
    - allow, deny: 以 allow 优先处理，但没有写入规则的则默认为 deny
- DirectoryIndex index.html index.html.var: 首页文件的文件名设置

### 20.2.3. PHP 的默认参数修改

1、Apache 将一些重要模块拆出来放置到 /etc/httpd/conf.d/\*.conf 文件中，PHP 模块配置文件为 /etc/httpd/conf.d/php.conf

2、PHP 安全方面的设定

- PHP 配置文件其实是 /etc/php.ini
- register\_globals=Off: 若设置为 On 则容易受到攻击
- log\_errors=On
- ignore\_repeated\_errors=On
- ignore\_repeated\_source=On
- 以上三个设置值决定是否将 PHP 程序的错误记录起来
- display\_errors=Off
- display\_startup\_errors=Off
- 以上两个设置值决定当程序发生问题时，是否要在浏览器上面显示相关的错误信息

3、PHP 提供上传容量限制 (/etc/php.ini)

- post\_max\_size=20M
- file\_uploads=On

- upload\_max\_filesize=16M
- memory\_limit=128M

#### 20.2.4. 启动 WWW 服务于测试 PHP 模块

- 1、systemctl restart httpd.service
- 2、在浏览器中输入"http://<本机的 IP>"
- 3、vim /var/www/html/phpinfo.php  
    <?php phpinfo(); ?>
  - http://<本机 IP>/phpinfo.php

#### 20.2.5. MySQL 的基本设定

- 1、启动 MySQL(设定 MySQL root 密码与添加 MySQL 用户账号)
  - MySQL 默认监听的端口在 port3306
  - MySQL 数据库软件也是个多用户的操作环境，登录时的 root 用户与 Linux 的 root 无关系哦
  - systemctl restart mysql.service
  - systemctl enable mysql.service
  - **mysqladmin -u root -p password** <==注意这里的 password 是关键字
    - Enter password: <==输入旧密码
    - New password: <==输入新密码
    - Confirm new password: <==再次输入密码
  - mysql -u root -p
- 2、效率调优/etc/my.cnf
  - 由于 MySQL 这个数据库系统如果在很多用户同时连接时，可能会造成某些效率方面的瓶颈
- 3、MySQLroot 密码遗忘的紧急处理
  - 如果你的数据库内容并不是很重，删除也无所谓的话(测试中)，那么可以将 MySQL 关闭后，将/var/lib/mysql/\*这个目录内的数据删除掉，然后重启 MySQL，那么 MySQL 数据库会重建，root 也没有密码了

#### 20.2.6. 防火墙设置与 SELinux 规则的放行

- 1、如果是小型网站，Apache 是连接到本机的 MySQL，并没有开放给外部的用户来连接数据库
  - 因此不要将 3306 放行给因特网连接，除非你真的知道你要给其他的服务器读取你的 MySQL
  - 因此只需要开放 80 端口即可
- 2、放行 80 端口
  - firewall-cmd --zone=public --add-port=80/tcp --permanent
  - firewall-cmd --zone=public --add-port=80/udp --permanent
  - firewall-cmd --reload
- 3、解决 SELinux 的规则放行问题
  - getsebool -a | grep httpd
  - setsebool -P httpd\_can\_network\_connect=1

## 20.2.7. 开始网页设计及安装架站软件，如 phpBB3

### 1、注意事项

- 默认的首页目录在/var/www/html/，应该将所有的 WWW 数据都搬到该目录下
- 注意你的资料权限(rwx 与 SELinux)，务必要让 Apache 的程序用户能够浏览
- 尽量将首页文件名定义为 index.html 或 index.php
- 如果想要建立在其他地方，那么需要修改 httpd.conf 中的 DocumentRoot 参数
- 不要将重要数据或者隐私数据放置到/var/www/html/首页内
- 如果需要安装一些 CGI 程序的话，建议你将它安装到/var/www/cgi-bin/下面，如此一来就不需要额外设置 httpd.conf 即可顺利启动 CGI 程序了

## 20.3. Apache 服务器的高级设定

### 20.3.1. 启动用户的个人网站(权限是重点)

#### 1、每台 WWW 服务器都有一个首页

- 每个用户都也可以有可以自己完全控制的首页
  - Apache 新版配置文件常常默认将这个功能取消掉
- ```
<IfModule mod_userdir.c>
    #UserDir disable    <==将这句注释掉
    UserDir www    <==默认是 public_html，这里改个名字而已
</IfModule>
```

2、**<未完成>**: 配置文件内容与书上的不一致

- cd /home/liuye
- mkdir www
- chmod 755 www
- chmod 711 /home/liuye
- cd www
- echo "Test your home" >> index.html
- setsebool -P httpd_enable_homedirs=1
- restorecon -Rv /home/
- 此时输入"http://<主机名>/~liuye"，找不到对应资源!!!
- cd var/www/html
- ln -s /home/liuye/www liuye
- 此时输入"http://<主机名>/liuye"，可以正确显示

20.3.2. 启动某个目录的 CGI (perl) 程序执行权限

1、如果想要 Apache 可执行 perl 之类的网页程序，就需要安装一些额外的模块才行，其中 mod_perl 与 mod_python 这两个软件建议安装(软件找不到???)

- yum install -y epel-release
- yum install -y mod_perl

2、利用新目录下的 Options 参数设定

- vim /etc/httpd/conf/httpd.conf
- 找到

```

#AddHandler cgi-script .cgi
● 改为
AddHandler cgi-script .cgi .pl
<Directory "/var/www/html/cgi">
    Options +ExecCGI
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
➤ mkdir /var/www/html/cgi
➤ vim /var/www/html/cgi/helloworld.pl
#!/usr/bin/perl
print "Content-type: text/html \r\n\r\n";
print "Hello, World.";
➤ chmod a+x /var/www/html/cgi/helloworld.pl

```

3、使用 ScriptAlias 的功能

20.3.3. 找不到网页时的显示信息通知

1、输入一个 URL 后，如果该目录下没有 index.???时

- 如果你的 Options 里面有设置 Indexes 的话，那么该目录下所有文件都会被列出来，提供类似 FTP 的连接页面
- 如果没有指定 Indexes 的话，那么错误信息就会被显示出来

2、步骤

- vim /etc/httpd/conf/httpd.conf
#ErrorDocument 404 /missing.html <==找到该句，将注释拿掉
- systemctl restart httpd.service
- 写一个/var/www/html/missing.html
- 在本机测试成功，在其他机器上失败，但是可以直接访问 missing.html

20.3.4. 浏览权限的设定操作(Order、Limit)

1、iptables 和 firewall 仅能一口气开放或整个拒绝，无法针对 WWW 的内容来放行

2、可以通过 Apache 内建的 Order 项目来处理

- Order deny,allow: 以 deny 优先处理，但没有写入规则的默认为 allow，常用于拒绝所有，开放特定条件

Order deny,allow

Deny from all

allow from 1.1.1.1

- 由于 all 里面也包含 1.1.1.1，因此重复了，所以这个 1.1.1.1 为默认值即 allow

- Order allow, deny: 以 allow 为优先处理，但没有写入规则的默认为 deny，常用于开放所有，拒绝特定条件

Order allow,deny

Allow from all

Deny from 1.1.1.1

- 由于 all 里面也包含 1.1.1.1，因此重复了，所以这个 1.1.1.1 为默认值即 deny
 - 如果 allow 与 deny 规则有重复，则以默认的情况(Order 的规范)为主
- 3、如果想让某个网络或者是 IP 无法浏览的话，最好利用 iptables 来处理，如果仅是某些重要目录不想让别人查阅的话，那么可以利用 allow, deny, Order 的设置来配置
- 4、另外，还可以用 Limit 这个设置值
- 允许进行 GET, POST 与 OPTIONS
- ```
<Limit GET POST OPTIONS>
 Order allow,deny
 Allow from all
</Limit>
```
- 不允许进行除此之外的其他操作
- ```
<LimitExcept GET POST OPTIONS>
    Order deny,all
    Deny from all
</LimitExcept>
```

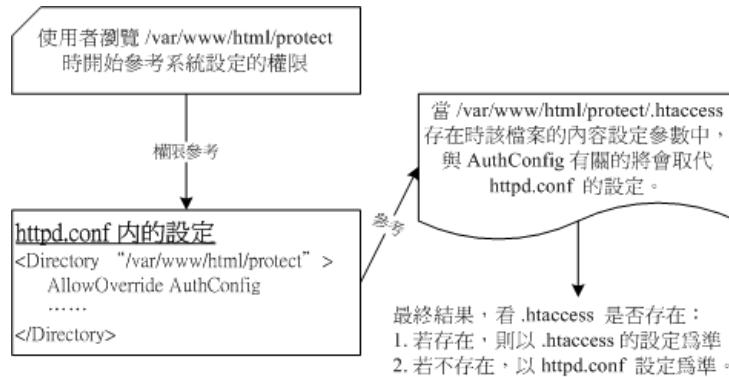
20.3.5. 服务器状态说明网页

- 1、我们可以通过 Apache 提供的特别功能来查询主机目前的状态，那就是 mod_status 模块，默认是关闭的，需要修改配置文件让其启动才行
- 2、<未完成>：找不到该模块，并且配置文件也不匹配

20.3.6. .htaccess 与认证网页设定

- 1、由于 Order 与 Limit 主要是针对 IP 网络或者主机名来管理的，如果客户端使用拨号方式取得 IP，那么 IP 会一直变动，那么这两个配置项就失效了
 - 2、此时如果能够使用密码保护的方式，让用户可以输入账号/密码即可取得浏览的权限，那么客户端就不用受到 Order 的 allow, deny 的限制了
 - 3、认证的网页如何处理
 - 建立受保护的目录
 - 设置 Apache 所需参数
 - 建立密码文件
 - 重启 Apache
- 4、如果我们能够通过外部文件来取代设置在 httpd.conf 内的参数，会比较好，而且该文件设置能够立即生效，不需要重启 Apache
- 通过 httpd.conf 的 AllowOverride 参数，配合 .htaccess 这个文件的设置就可以实现上述所预想的功能
 - **主配置文件 httpd.conf 的修改：**
 - 必须要在 httpd.conf 这个配置文件中先以 AllowOverride 指定某个目录下的 .htaccess 有哪些能够进行取代的参数，一般有 AuthConfig、Options 等
 - 考虑到系统数据的安全，建议提供 AuthConfig 的项目，设置完毕后重启 Apache
 - **.htaccess 放置的目录**

- 在受保护的目录下面务必要存在`.htaccess`这个文件，通过这个文件即可修改`httpd.conf`内的设置
- **.htaccess 的修改**
- `.htaccess` 设置完立刻生效，不需要重启 Apache，因为该文件的内容是“当客户端浏览器到该目录时，该文件才会被用来取代原有的设置”



图表 20-1 .htaccess 与主要配置文件 httpd.conf 的相关性

5、建立保护目录的数据

- `mkdir /var/www/html/protect`
- `vim /var/www/html/protect/index.html`

6、以 root 的身份处理 httpd.conf 的设置数据

- `vim /etc/httpd/conf/httpd.conf`

```
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
    Satisfy All
</Files>
```

```
<Directory "var/www/html/protect">
    AllowOverride AuthConfig
    Order allow,deny
    Allow from all
</Directory>
```

- `systemctl restart httpd.service`

7、建立保护目录下的`.htaccess`文件：只要有权限建立者即可进行

- `cd /var/www/html/protect`
- `vim .htaccess`

```
AuthName "Protect test by .htaccess"      <==提示信息
AuthType Basic    <==认证类型
AuthUserFile /var/www/apache.passwd      <==保护目录所使用的账号密码配置文件，不要放在 Apache 可以浏览的目录内
require user test    <==可用的账号，如果要让密码文件内的用户都能登录，改成"require valid-user"即可
```

8、建立密码文件 htpasswd(只要有权限即可执行)

- `htpasswd -c /var/www/apache.passwd test` <==添加账号，-c 可能是创建文件吧

- htpasswd /var/www/apache.passwd test1 <==添加账号，将.htaccess 改成 "require valid-user"后 test1 才能访问

20.3.7. 虚拟主机的设定(重要)

- 1、虚拟主机可以让你的一台 Apache 看起来像有多个"主站首页"
- 2、什么是虚拟主机(Virtual Host)
 - 让你的一台服务器上面，有好多个"主网页"存在，也就是说，硬件实际只有一台主机，但是由网站网址上来看，似乎有多台主机存在的样子
 - 可以让你的多个主机名对应到不同的主网页目录(DocumentRoot 参数)
- 3、假设的大前提：同一个 IP 有多个主机名
 - 如何拥有多个主机名
 - 向 ISP 申请多个合法的主机名，而不自己假设 DNS
 - 自行设置经过合法授权的 DNS 主机来设置自己所需的主机名
- 4、一个架设练习
 - mkdir /var/www/www
 - mkdir /var/www/ftp
 - echo "www.ittc.liuye" > /var/www/www/index.html
 - echo "ftp.ittc.liuye" > /var/www/ftp/index.html
 - 开始编辑配置文件，这里用额外的文件来配置
 - vim /etc/httpd/conf.d/virtual.conf
 - #规定任何接口的 port80 所指定的虚拟主机
NameVirtualHost *:80

#设置两个目录的访问权限以及可执行的操作等

```
<Directory "/var/www/www">
    Options FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

```
<Directory "/var/ftp">
    Options FollowSymLinks Indexes
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

#针对三台主机的 DocumentRoot 进行设置

```
<VirtualHost *:80>
    ServerName linux.ittc.liuye
    DocumentRoot /var/www/html
</VirtualHost>
```

```
<VirtualHost *:80>
    ServerName www.ittc.liuye
    DocumentRoot /var/www/www
    CustomLog /var/log/httpd/www.access_log combined
</VirtualHost>
```

```
<VirtualHost *:80>
    ServerName ftp.ittc.liuye
    DocumentRoot /var/www/ftp
</VirtualHost>
```

- 在虚拟主机的设置上还有许多可用功能，最低的限度是需要有个 `ServerName` 以及 `DocumentRoot` 两个
- 使用了虚拟机主机后，原本的主机名(`linux.ittc.liuye`)也要同时写入虚拟主机的对应中，否则将丢失了
- 在 `www.ittc.liuye` 这个主机中多了个 `CustomLog`，表示任何向 `www.ittc.liuye` 要求数据的记录都会改写入 `/var/log/httpd/www.access_log` 中而不是默认的 `/var/log/httpd/access_log`，但是这个新建的日志文件必须要加入 `logrotate` 的管理中，否则日志文件会越来越大

5、虚拟主机的常见用途

- 主机代管
- 服务器数据备份系统
- 将自己的数据分门别类

20.4. 日志文件分析以及 PHP 强化模块

20.4.1. PHP 强化模块(eaccelerator)与 Apache 简易性能测试

1、PHP 网页程序标榜的是速度快，不过因为 PHP 毕竟是将一些可用函数先编译成为模块，然后当网页使用到 PHP 程序的时候，再呼叫 PHP 模块来实现程序所需要的行为

2、如果可以将 PHP 程序预先转换成为可直接执行的 `binary file`，就可以进一步加快速度，这东西就叫与编译器

- 其中有一套软件称为 `eaccelerator`，它可以将你的 PHP 程序与 PHP 核心及相关函数库预先编译后暂存下来，已提供未来使用时可以直接执行，加上他可以优化 PHP 程序，因此可以让 PHP 网页速度增快不少

3、步骤

- 下载源码包(<http://eaccelerator.net>)
- 确保安装了 `php-devel`、`autoconf`、`automake`、`m4`、`libtools` 等软件
- `yum install -y php-devel autoconf automake m4 libtools`
- `tar` 解压后，`cd` 进入解压目录
- `phpize`
- `./configure --enable-eaccelerator=shared \`
`--with-php-config=/usr/bin/php-config`
- `make`

```

➤ make install
➤ echo "/usr/lib64/php/modules/" >> /etc/ld.so.conf.d/php.conf
➤ ldconfig
➤ vim /etc/php.ini, 在最后加上如下内容
;http://eaccelerator.net/      ;
;2016/12/03 Liuye            ;
extension="eaccelerator.so"
eaccelerator.shm_size="16"
eaccelerator.cache_dir="/tmp/eaccelerator"
eaccelerator.enable="1"
eaccelerator.optimizer="1"
eaccelerator.check_mtime="1"
eaccelerator.debug="0"
eaccelerator.filter=""
eaccelerator.shm_max="0"
eaccelerator.shm_ttl="0"
eaccelerator.shm_prune_period="0"
eaccelerator.shm_only="0"
eaccelerator.compress="1"
eaccelerator.compress_level="9"
➤ mkdir /tmp/eaccelerator
➤ chmod 777 /tmp/eaccelerator
➤ systemctl restart httpd.service

```

4、测试<ab>

```

➤ ab [-dSk] [-c number] [-n number] 网页文件名
➤ -d 不要显示 saved table 的百分比数据
➤ -k: KeepAlive, 加入-k 才有这样的功能测试
➤ -S: 不显示长信息, 仅显示类似/min/avg/max 的简短易懂的信息
➤ -c: 同时有多少个"同时连接"的设置(可理解为同时连接的 IP)
➤ -n: 同一个连接建立几个要求通道(可理解为同一个 IP 要求几条连接)
➤ ab -dSk -c100 -n100 http://localhost/phpinfo.php

```

20.4.2. syslog 与 logrotate

1、Apache 日志文件主要记录两个内容

```

➤ /var/log/httpd/access_log: 客户端正常要求的记录信息
➤ /var/log/httpd/error_log: 用户错误要求的数据, 包括服务器设置错误的信息等

```

20.4.3. 日志文件分析软件: webalizer

1、<未完成>: yum 没有

20. 4. 4. 日志文件分析软件: awstats

1、<未完成>: 呵呵

20. 5. 建立连接加密网站(https)及防整站下载脚本

1、http 这个通信协议是明码传送数据, https 是加密传输, 加密方式通过 ssh

20. 5. 1. 拥有自制证书的 https

1、建立证书文件

2、<未完成>: 呵呵

Chapter 21. 文件服务器之三：FTP 服务器

21.1. FTP 的数据传输原理

1、FTP(File Transfer Protocol)是相当古老的传输协议之一，它最主要的功能是在服务器与客户端之间进行文件的传输，该协议使用的是明文传输方式

21.1.1. FTP 功能简介

1、FTP 服务器的功能除了单纯地进行文件的传输与管理之外，依据服务器软件的配置架构，它还可以提供以下几个主要的功能

- 不同等级的用户身份：user、guest、anonymous
 - FTP 服务器在默认的情况下，依据用户登录的情况分为三种不同的身份，分别是实体用户(real user)、访客(guest)、匿名用户(anonymous)
 - 这三种用户能够使用的在线命令不同，在系统上面的权限差异很大
- 命令记录与日志文件记录
 - FTP 可以利用系统的 syslog(rsylog)来进行数据的记录，而记录的数据包括了用户曾经使用过的命令与用户传输数据的记录
- 限制用户活动的目录
 - 用户的工作范围局限在用户主目录下面，FTP 可以限制用户仅能在自己的用户主目录当中活动
 - chroot

21.1.2. FTP 的工作流程与使用到的端口

1、FTP 的传输使用的是 TCP 数据包协议，TCP 在建立连接前会进行三次握手，FTP 服务器使用了两个连接，分别是**命令通道**与**数据流通道(ftp-data)**，这两个连接都需要经过三次握手

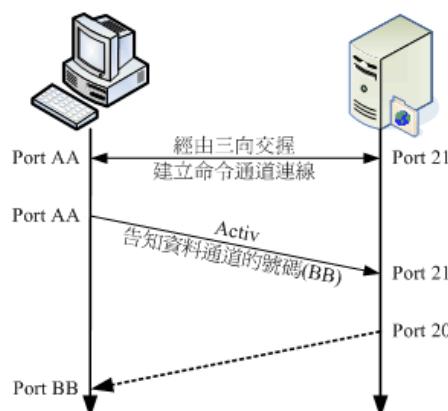


图 21-1 FTP 服务器的主动式连接示意图

2、连接流程

- 建立命令通道的连接
 - 客户端会随机取一个大于 1024 以上的端口(port AA)来与 FTP 服务器端的 port21 实现连接
 - 连接后客户端便可通过这个连接来对 FTP 服务器执行命令，查询文件名、下载、上传等等命令都是通过这个通道执行的
- 通知 FTP 服务器端使用 Active 且告知连接的端口号

- FTP 服务器的端口号 21 主要在命令的执行，但是当牵涉到数据流时，就不是使用这个连接了
- 客户端在需要数据的情况下，会告知服务器端要用什么方式来连接，如果是主动式(Active)连接时，客户端会先随机启用一个端口，且通过命令通道告知 FTP 服务器这个消息，并等待 FTP 服务器的连接
 - FTP 服务器主动向客户端连接
 - FTP 服务器由命令通道了解客户端的需求后，会主动地由 Port20 向客户端的 port BB 连接，这个连接也需要三次握手
 - 此时 FTP 的客户端与服务器端会建立两条连接，分别在命令的执行与数据的传递
 - 默认的 FTP 服务器端使用的主动连接端口就是 port20
- 注意，有传输数据的需求时，那么数据传输通道才会建立，并不是客户端连接到 FTP 服务器时就建立的

3、主动式连接使用到的端口号

- 命令通道的 ftp(默认为 port21)，客户端主动连接到 FTP 服务器
- 数据传输的 ftp-data(默认为 port20)，FTP 服务器主动连接到客户端
- **这样的情况在服务器与客户端两者同为 Public IP 时是没有问题的，但是如果在防火墙或者 NAT 服务器后面呢**

4、在主动连接的 FTP 服务器与客户端之间具有防火墙的

- 1) 用户与服务器间命令通道的建立
 - 因为 NAT 会主动记录由内部送往外部的连接信息，而由于命令通道是由客户端向服务器端连接的，因此这一条连接时可以顺利建立的
 - 2) 用户与服务器间数据通道建立的通知
 - 同样，客户端会先启用 port BB，并通过命令通道告知 FTP 服务器，等待服务器端的主动连接
 - 3) 服务器主动连接到 NAT 等待转递至客户端的连接问题
 - 由于通过 NAT 的转换后，FTP 服务器只能得知 NAT 的 IP 而不是客户端的 IP，因此 FTP 服务器会以 port20 主动向 NAT 的 port BB 发送主动连接的要求，但 NAT 并没有启动 port BB 来监听 FTP 服务器的连接
- 问题在于：FTP 的主动式连接中，NAT 将会被视为客户端，但 NAT 其实并非客户端
 - 解决方法
 - 使用 iptables 提供的 FTP 检测模块：这个模块会主动分析目标的 port21 的连接信息，所以可以得到 port BB 的资料，若此时接收到 FTP 服务器的主动连接，就能够将该数据包导向正确的后端主机了(必须是默认的 port21 端口哦，若在非默认端口那么该模块是没用的)
 - 客户端选择被动式(Passive)连接模式：被动式就是由客户端向服务器发起连接

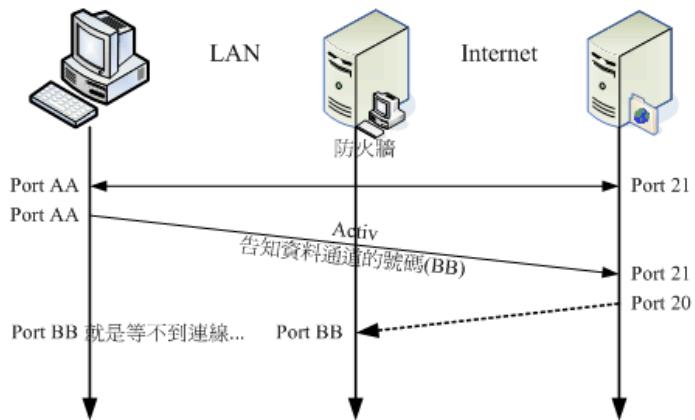


图 21-2 FTP 客户端与服务器端连接中间具有防火墙的连接状态

21.1.3. 客户端选择被动式连接模式

1、被动式连接流程

- 用户与服务器建立命令通道
- 客户端发起 PASV 的连接要求
 - 当使用数据通道的命令时，客户端可通过命令通道发出 PASV 的被动式连接要求，并等待服务器的回应
- FTP 服务器启动数据端口，并通知客户端连接
 - 如果你使用的 FTP 服务器是能够处理被动式连接的，此时 FTP 服务器会先启动一个监听端口，这个端口号号码是随机的，也可以自定义某一范围的端口
 - 然后 FTP 服务器会通过命令通道告知客户端该已经启动的端口，并等待客户端的连接
- 客户端随机取用大于 1024 的端口进行连接
 - 然后你的客户端会随机取用一个大于 1024 的端口号来进行对主机的 port PASV 连接

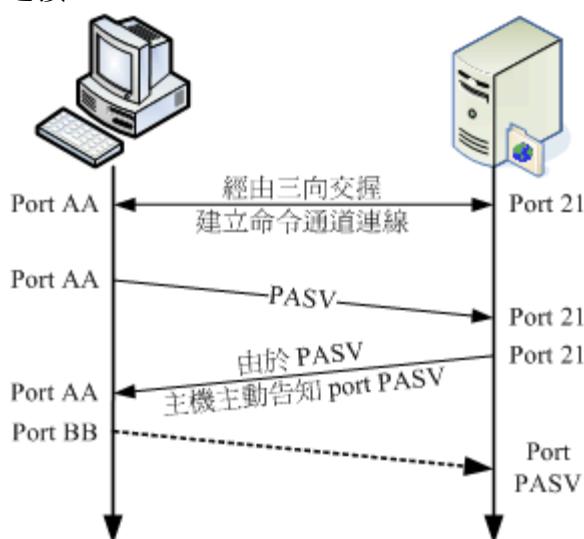


图 21-3 FTP 的被动式数据流连接流程

21.1.4. FTP 的安全性问题与替代方案

1、在 FTP 上面传送的数据很可能被窃取，因为 FTP 是明文传输的，而且某些 FTP

服务器软件的安全历史问题也很严重

2、由于 SSH 技术的产生，目前我们可以拥有较为安全的 FTP，就是 SSH 提供的 sftp 这个 Server，其最大的优点就是：在上面传输的数据是经过加密的，所以在因特网上时比较安全一些

3、假设 FTP 网站的注意事项

- 随机更新到最新版本的 FTP 软件，并随时注意漏洞信息
- 善用 iptables 来规定可以使用的 FTP 网络
- 善用 TCP_Wrappers 来规范可以登录的网络
- 善用 FTP 软件的设置来限制使用你的 FTP 服务器的用户的不同权限
- 使用 Super daemon 来管理你的 FTP 服务器
- 随时注意用户的用户主目录以及匿名用户登录的目录的文件权限
- 若不对外公开的话，或许也可以修改 FTP 的 port
- 也可以使用 FTPs 这种加密的 FTP 功能

21.1.5. 开放什么身份的用户登录

1、开放实体用户的情况(Real User)

- 使用替代的 FTP 方案比较好：由于实体用户本来就可以通过网络连接到主机来进行工作(例如 SSH)，因此没有必要特别开放 FTP 的服务
- 限制用户能力，如 chroot 与/sbin/nologin 等：如果确定要让实体用户利用 FTP 服务器的话，那么可能需要让某些系统账号无法登陆 FTP 才行，例如 bin/apachedeng，最简单的方法就是通过 PAM 模块来处理

2、访客身份(Guest)

- 在通常会建立 guest 身份的案例中，多半是由于服务器提供了类似个人 Web 首页的功能给一般身份用户，这个时候将用户的身份压缩成 guest，并且将他的可用目录设置好，即可提供用户一个方便的使用环境了
- 常见的建议
 - 仅提供需要登录的账号即可，不需要提供系统上面所有人均可登录的环境
 - 在服务器的设置当中，我们需要针对不同的访客给他们不一样的用户主目录，而这个用户主目录与用户的权限设置需要相符合
 - 针对这样的用户身份，需要设置较多的限制，包括上传(下载)文件数目与硬盘容量的限制、连接登录的时间限制、许可使用的命令限制，例如不许使用 chmod 等等

3、匿名登录用户(anonymous)

- 提供匿名用户进入因特网实在不是个好主意，因为每个人都可以下载你的数据，但是对于学校等单位要共享给全体同学一些软件资源时，这是一个比较好的解决方案
- 如果真要开放匿名登录，很多限制都是必须的，比如
 - 允许的工作命令要减少很多，几乎就不允许匿名用户使用命令
 - 限制文件传输的数量，尽量不要允许上传数据的设置
 - 限制匿名用户同时登录的最大连接数量，可以控制盗连

21.2. vsftpd 服务器基础设置

1、vsftpd 的全名是"Very Secure FTP Daemon", vsftpd 最初的发展理念就是在构建一个以安全为重心的 FTP 服务器

21.2.1. 为何使用 vsftpd

1、**vsftpd 针对操作系统的程序的权限(privilege)概念来设计**, 系统上面所执行的程序都会引发一个程序, 我们称它为 PID, 这个 PID 在系统上面能进行的任务与它拥有的权限有关, 也就是说 PID 拥有的权限等级越高, 它能够进行的任务就越多

2、如果触发这个 PID 的程序有漏洞而导致被网络黑客所攻击而取得 PID 使用权时, 那么网络黑客将会取得这个 PID 拥有的权限, **因此近来发展的软件都会尽量将服务取得的 PID 权限降低, 使得该服务即使不小心被入侵了, 入侵者也无法得到有效的系统管理权限, 这样就会让系统较为安全, vsftpd 就是基于这种想法设计的**

3、除了 PID 方面的权限之外, vsftpd 也支持 chroot 这个函数的功能, chroot 顾名思义就是"change root directory"的意思, 如果以匿名身份登录 FTP 服务器, 通常会被限定在/var/ftp 目录下工作, 而你看到的根目录就是/var/ftp, 至于其他目录是无法看见的, 这样系统就会比较安全了

4、vsftpd 的特点

- vsftpd 的服务器启动者身份为一般用户, 所以对于 Linux 系统的权限较低, 对于 Linux 系统的危害就相对降低了, 此外, vsftpd 也利用 chroot()这个函数进行改换根目录的操作, 使得系统工具不会被 vsftpd 这个服务所误用
- 任何需要具有较高执行权限的 vsftpd 命令均以一个特殊的上层程序所控制, 该上层程序享有的教导执行权限功能已经被限制得相当低, 并以不影响 Linux 本身的安全为准
- 绝大部分 FTP 会使用到的额外命令功能(dir、ls、cd 等)都已经被整合到 vsftpd 主程序当中了, 因此理论上 vsftpd 不需要使用到额外的系统提供的命令, 所以在 chroot 的情况下, vsftpd 不但可以顺利工作, 而且不需要额外功能对于系统来说也比较安全
- 所有来自客户端且想要使用这个上层程序所提供的较高执行权限的 vsftpd 命令需求, 均被视为不可信任的要求来处理, 必须要经过相当程度的身份确认后, 方可利用该上层程序的功能, 例如 chown、login 等操作
- 上层程序中, 依然使用 chroot 的功能来限制用户的执行权限

21.2.2. 所需要的软件以及软件结构

1、vsftpd 所需要的软件只有一个, 那就是 vsftpd

- yum install vsftpd

2、相关文件目录

- /etc/vsftpd/vsftpd.conf
 - 严格来说, 整个 vsftpd 的配置文件就只有这个文件
 - 这个文件的设置是以 bash 的变量设置相同的方式来处理的, 也就是"参数=设置值", **注意等号两边不能有空白**
- /etc/pam.d/vsftpd

- vsftpd 使用 PAM 模块时的相关配置文件，主要用来作为身份认证之用，还有阻挡一些用户身份的功能
- /etc/vsftpd/ftpusers
 - PAM 模块(/etc/pam.d/vsftpd)所指定的那个无法登陆的用户配置文件
 - 只要将不想让他登陆的 FTP 账号写入这个文件即可
- /etc/vsftpd/user_list
 - 这个文件能否生效与 vsftpd.conf 内的两个参数有关，分别是 userlist_enable 与 userlist_deny
 - /etc/vsftpd/ftpusers 是 PAM 模块的阻挡访问设置项目，那么 /etc/vsftpd/user_list 则是 vsftpd 自定义的阻挡访问项目
 - 该文件与/etc/vsftpd/ftpusers 几乎一模一样
 - 该文件是否其作用与 vsftpd.conf 的"userlist_deny={YES/NO}"有关
- /etc/vsftpd/chroot_list
 - 该文件默认不存在，
 - 该文件主要功能时可以将某些账号的用户 chroot 建立在他们的默认用户主目录下
 - 该文件是否生效与 vsftpd.conf 的 chroot_list_enable 与 chroot_list_file 两个参数有关
 - 如果想要将某些实体用户限制在他们的用户主目录下而不允许到其他目录去，可以设置这个项目
- /usr/sbin/vsftpd
 - vsftpd 的主要执行文件
- /var/ftp/
 - 这是 vsftpd 默认匿名用户登录的根目录，其实与 ftp 这个账号的用户主目录有关

21.2.3. vsftpd.conf 配置值说明

1、以下为与服务器环境比较相关的设置值

- connect_from_port_20=YES(NO): 主动式连接使用的 FTP 服务器的 port 号
- listen_port=21: vsftpd 使用的命令通道 port，如果你想要使用非正规的端口号，可在这个设置项修改，这个设置项仅适合一 stand alone 方式来启动(对于 super daemon 无效)
- dirmessage_enable=YES(NO): 当用户进入某个目录时，会显示该目录需要注意的内容，显示的文件默认是.message
- message_file=.message: 当 dirmessage_enable=YES 时，可以设置这个项目让 vsftpd 寻找该文件来显示信息
- listen=YES(NO): 若设置为 YES，表示 vsftpd 是以 stand alone 的方式来启动，默认是 NO，所以 CnetOS 将它改为 YES，这样才能使用 stand alone 的方式来唤醒
- pasv_enable=YES(NO): 支持数据流的被动式连接模式(passive mode)，一定要设为 YES
- use_localtime=YES(NO): 是否使用本地时间，vsftpd 默认使用 GMT 时间(格林尼治时间)，因此默认的 FTP 的文件日期会比中国晚 8 小时，建议改为 YES

- `write_enable=YES(NO)`: 是否允许用户上传数据
- `connect_timeout=60`: 单位秒, 在数据连接的**主动式**连接模式下, 发出的信号 60 秒内得不到客户端响应, 则不等待并强制断线
- `accept_timeout=60`: 当以**被动式** PASV 来进行数据传输时, 如果服务器启用 `passive port` 并等待 Client 超过 60 秒无回应, 那么就强制断线, 与 `connect_timeout` 类似, 但是一个管理主动, 一个管理被动
- `date_connection_timeout=300`: 如果服务器与客户端的数据连接已经建立(无论主动还是被动), 但是可能由于线路问题导致 300 秒内还是无法顺利地完成数据传送, 那客户端的连接就会被我们的 vsftpd 强制剔除
- `idle_session_timeout=300`: 如果用户在 300 秒内没有命令操作, 强制脱机
- `max_clients=0`: 如果 vsftpd 是以 stand alone 方式启动的, 那么这个设置项目可以设置同一时间最多有多少 Client 可以同时连上 vsftpd, 限制使用 FTP 的用量
- `max_per_ip=0`: 同一个 IP 同一时间可允许多少连接
- `pasv_min_port=0、pasv_max_port=0`: 0 表示随机选取不受限制
- `ftpd_banner=`一些文字说明: FTP 客户端软件上会显示的说明文字
- `banner_file=/path/file`: 这个项目可以指定某个纯文本作为用户登录 vsftpd 服务器时所显示的欢迎字眼

2、与实体用户较相关的设置值

- `guest_enable=YES(NO)`: 若为 YES, 则任何实体账号, 均会被假设成为 guest, 因此默认是不开放的
- `guest_username=ftp`: 在 `guest_enable` 指定为 YES 时生效, 指定访客身份
- `local_enable=YES(NO)`: 设为 YES 时, 在 /etc/passwd 内的账号才能以实体用户的方式登录 vsftpd 服务器
- `local_max_rate=0`: 实体用户的传输速度限制, 单位为 bytes/second, 0 为不限制
- `chroot_local_user=YES(NO)`: 是否被更改"根目录"
- `chroot_list_enable=YES(NO)`: 是否启用 chroot 写入列表的功能
- `chroot_list_file=/etc/vsftpd.chroot_list`: 当 `chroot_list_enable` 为 YES 时, 生效
- `userlist_enable=YES(NO)`: 是否借助 vsftpd 的阻挡机制来处理某些不受欢迎的账号
- `userlist_deny=YES(NO)`: 当 `userlist_enable=YES` 时才生效, 若为 YES, 则当用户账号被列入某个文件时, 在该文件内的用户将无法登录 vsftpd 服务器
- `userlist_file=/etc/vsftpd/user_list`: 若 `userlist_deny=YES` 时, 该文件生效

3、匿名用户登录的设置值

- `anonymous_enable=YES(NO)`: 设置为允许 anonymous 登录 vsftpd 主机, 默认 YES, **以下设置值只有在 anonymous_enable=YES 时才生效**
- `anon_world_readable_only=YES(NO)`: 仅允许 anonymous 具有下载可读文件的权限, 默认 YES
- `anon_other_write_enable=YES(NO)`: 是否允许 anonymous 具有除了写入之外的权限, 包括删除与修改服务器上的文件及文件名等权限, 默认是 NO, 如果设置为 YES, 那么开放给 anonymous 写入的目录亦需要调整权限, 让 vsftpd 的 PID 拥有者可以写入才行

- `anon_mkdir_write_enable=YES(NO)`: 是否让 `anonymous` 具有建立目录的权限, 默认为 NO, 如果设置为 YES, 那么只有 `anon_other_write_enable=YES` 时才生效
- `anon_upload_enable=YES(NO)`: 是否让 `anonymous` 具有上传数据的功能, 默认 NO, 若为 YES, 则当 `anon_other_write_enable=YES` 才生效
- `deny_email_enable=YES(NO)`: 将某些特殊的 E-mail address 阻挡住
- `banned_email_file=/etc/vsftpd/banned_emails`: 当 `deny_email_enable=YES` 时, 利用这个项目来规定哪个 Email address 不能登录 vsftpd, 在文件内, 一行输入一个 E-mail address 即可
- `no_anon_password=YES(NO)`: 当设置为 YES 时, 表示 `anonymous` 将会略过密码检验步骤, 而直接进入 vsftpd 服务器内, 因此一般默认 NO
- `anon_max_rate=0`: 限制 `anonymous` 的传输速度, 单位 bytes/s
- `anon_umask=007`: 限制 `anonymous` 上传文件的权限

4、关系系统安全方面的设置值

- `ascii_download_enable=YES(NO)`: 如果设为 YES, 则 Client 就优先使用 ASCII 格式下载文件
- `ascii_upload_enable=YES(NO)`: 与上一个类似, 针对上传, 默认 NO
- `one_process_model=YES(NO)`: 当设置为 YES 时, 表示每个建立的连接都会拥有一个 process 在负责, 可以提高 vsftpd 效率, 不过除非系统比较安全, 而且硬件配备比较高, 否则建议 NO
- `tcp_wrappers=YES(NO)`: 支持 Wrappers
- `xferlog_enable=YES(NO)`: 设置为 YES 时, 用户上传与下载文件都会被记录下来, 记录的文件与下一个设置项目有关
- `xferlog_file=/var/log/xferlog`: 当 `xferlog_enable=YES` 时, 该设置值生效, 即日志文件文件名
- `xferlog_std_format=YES(NO)`: 是否设置为 wu-ftp 相同的日志文件格式, 默认认为 NO, 如果有 wu-ftp 日志文件的分析软件, 这里才需要设置 YES
- `dual_log_enable=YES`、`vsftpd_log_file=/var/log/vsftpd.log`: 除了 /var/log/xferlog 的 wu-ftp 格式日志文件之外, 还可以具有 vsftpd 独特日志文件格式, 若服务器不忙碌, 则可以同时记录两个日志 (`/var/log{vsftpd.log,xferlog}`)
- `nopriv_user=nobody`: vsftpd 默认以 nobody 作为此服务的执行者的权限, 因为 nobody 权限相当低, 因此即使被入侵, 入侵者仅能取得 nobody 权限
- `pam_service_name=vsftpd`: PAM 模块的名称

21.2.4. vsftpd 启动的模式

1、vsftpd 可以使用 stand alone 或 super daemon 的方式来启动, CentOS 默认以 stand alone 来启动

2、利用 CentOS 的 systemd 来启动(stand alone)

- `systemctl restart vsftpd.service`

3、设置以 super daemon 来启动

- `vim /etc/vsftpd/vsftpd.conf`
- `listen=NO`

- yum install xinetd
- vim /etc/xinetd.d/vsftpd
- 失败，无法启动 xinetd

21.2.5. CentOS 的 vsftpd 默认值

1、查看/etc/vsftpd/vsftpd.conf， 默认值如下

- anonymous_enable=YES <==支持匿名用户登录
- local_enable=YES <==支持本地端实体用户登录
- write_enable=YES <==允许用户上传数据
- local_umask=022 <==建立新目录的权限
- dirmessage_enable=YES <==若木路下有.message，则会显示该文件内容
- xferlog_enable=YES <==启动日志文件记录，记录于/var/log/xferlog
- connect_from_port_20=YES <==支持主动式连接功能
- xferlog_std_format=YES <==支持 WuFTP 的日志文件格式
- listen=YES <==使用 stand alone 方式启动 vsftpd
- pam_service_name=vsftpd <==支持 PAM 模块管理
- userlist_enable=YES <==支持/etc/vsftpd/user_list 文件内的账号登录控制
- tcp_wrappers=YES <==支持 TCP Wrappers 的防火墙机制

2、于是 vsftpd 可以实现如下功能

- 可以使用 anonymous 这个匿名账号或其他实体账号登录
- anonymous 的用户主目录在/var/ftp，且无上传权限，已经被 chroot 了
- 实体用户主目录参考/etc/passwd，并没有被 chroot，可前往任何有权限可进入的目录中
- 任何于/etc/vsftpd/ftpusers 内存在的账号均无法使用 vsftpd(PAM)
- 可利用/etc/hosts.{allow|deny}来作为基础防火墙
- 当客户端有任何上传/下载信息时，会被记录到/var/log/xferlog 中
- 主动式连接的端口号为 20
- 使用格林尼治时间(GMT)

21.2.6. 针对实体账号的设定

1、希望有的功能

- 希望使用本地时间取代 GMT 时间
- 用户登录时显示一些欢迎信息的信息
- 系统账号不可登录主机，即 UID 小于 500 以下的账号
- 一般实体用户可以进行上传、下载，建立目录以及修改文件等操作
- 用户建立文件、目录的 umask 希望设置为 002
- 其他主机设置值保留系统默认值

2、修改主配置文件/etc/vsftpd/vsftpd.conf

anonymous_enable=NO

```
local_enable=YES
write_enable=YES
local_umask=022
userlist_enable=YES
```

```
userlist_deny=YES  
userlist_file=/etc/vsftpd/user_list
```

```
use_localtime=YES  
dirmessage_enable=YES  
xferlog_enable=YES  
connect_from_port_20=YES  
xferlog_std_format=YES  
listen=YES
```

```
pam_service_name=vsftpd  
tcp_wrappers=YES  
banner_file=/etc/vsftpd/welcome.txt
```

3、建立欢迎信息

➤ vim /etc/vsftpd/welcome.txt

4、建立限制系统账号登录的文件

➤ 针对系统账号来给予阻挡的机制，其实有两个文件，一个是 PAM 模块管的，一个是 vsftpd 主动提供的，在默认情况下，这两个文件是

- /etc/vsftpd/ftpusers
- /etc/vsftpd/user_list

5、实体账号的 SELinux 议题

➤ 默认情况下，CentOS 的 FTP 是不允许实体账号登录取得用户主目录数据的，这是因为 SELinux 的问题

➤ setsebool -P ftp_home_dir=1

6、对用户(包括未来新建用户)进行 chroot

➤ 建议让实体用户全部被 chroot，而允许不必 chroot 的账号才需要额外设置

➤ vim /etc/vsftpd/vsftpd.conf

```
chroot_local_user=YES  
chroot_list_enable=YES  
chroot_list_file=/etc/vsftpd/chroot_list
```

7、限制实体用户的总下载流量

➤ vim /etc/vsftpd/vsftpd.conf
local_max_rate=1000000

8、限制最大同时上线人数同一 IP 的 FTP 连接数

➤ vim /etc/vsftpd/vsftpd.conf
max_client=10
max_per_ip=1

9、建立严格的可使用 FTP 的账号列表

➤ vim /etc/vsftpd/vsftpd.conf
userlist_enable=YES
userlist_deny=NO
userlist_file=/etc/vsftpd/user_list

21.2.7. 仅有匿名登录的相关设置

1、一般这种设置是给类似大专院校的 FTP 服务器来使用的

- 使用本地时间，而非 GMT 时间
- 提供欢迎信息，说明可提供下载的信息
- 仅开放 anonymous 的登录，且不需要输入密码
- 文件传输限速 1Mbytes/second
- 数据的连接过程(不是命令通道)只要超过 60 秒没有响应，就强制 Client 断线
- 只要 anonymous 超过 10 分钟没有操作，就予以断线
- 最大同时上线人数限制为 50 人，且同一 IP 来源最大连接数为 5 人

2、默认的 FTP 匿名用户的根目录所在：ftp 账号的用户主目录

- 匿名用户默认登录的根目录是以 ftp 这个用户的用户主目录为主
- CentOS 默认的匿名用户根目录在 /var/ftp/ 中，且匿名登陆者在使用 FTP 时，他默认可以使用 ftp 这个用户身份的权限，只是被 chroot 到 /var/ftp/ 目录中

➤ mkdir /var/ftp/linux

➤ mkdir /var/ftp/gnu

➤ vim /etc/vsftpd/vsftpd.conf **重新设置配置文件**

anonymous_enable=YES

no_anon_password=YES

anon_max_rate=1000000

data_connection_timeout=60

idle_session_timeout=600

max_clients=50

max_per_ip=5

local_enable=NO

use_localtime=YES

dirmessage_enable=YES

xferlog_enable=YES

connect_from_port_20=YES

xferlog_std_format=YES

listen=YES

pam_service_name=vsftpd

tcp_wrappers=YES

banner_file=/etc/vsftpd/anon_welcome.txt

➤ systemctl restart vsftpd.service

➤ ftp localhost

● anonymous

● [Enter]

3、让匿名用户可以上传/下载自己的资料(权限开放最大)

➤ vim /etc/vsftpd/vsftpd.conf, **在上一小节的基础上，增加如下几行**

write_enable=YES

```
anon_other_write_enable=YES  
anon_mkdir_write_enable=YES  
anon_upload_enable=YES  
➤ mkdir /var/ftp/uploads  
➤ chown ftp /var/ftp/uploads  
➤ setsebool -P allow_ftpd_anon_write=1  
➤ setsebool -P allow_ftpd_full_access=1
```

4、让匿名用户仅具有上传权限，不可下载匿名用户上传的东西

- 学生提交作业，但是不能下载别人上传的文件
- vim /etc/vsftpd/vsftpd.conf

```
<删掉 anon_other_write_enable=YES>  
write_enable=YES  
anon_mkdir_write_enable=YES  
anon_upload_enable=YES  
chwon_uploads=YES  
chown_username=daemon
```

- 于是上传的文件将会被修改文件拥有者称为 **daemon** 这个用户，**ftp** 是无法读取 **daemon** 这个用户的数据的

5、被动式连接端口的限制

- 主动式连接比较好处理，因为都是通过服务器的 port20 对外主动连接，所以防火墙的处理比较简单
- 被动式连接比较麻烦，因为默认的 **FTP** 服务器会随机取几个没有在使用中的端口来建立被动式连接，那防火墙的设置就很麻烦
- 我们可以通过指定几个固定范围内的端口来作为 **FTP** 的被动式数据连接之用即可

```
pasv_min_port=65400  
pasv_max_port=65410
```

21.2.8. 防火墙设置

1、需要进行如下设置

- 开放 port21 给因特网使用
- 开放前一小节的 port65400-65410 端口给 Internet 使用

21.2.9. 常见问题与解决之道

1、无法连接成功

- 是否开放了 port21 端口
- 在/etc/hosts.deny 中，是否将 Client 的登录权限挡住了
- 在/etc/xinetd.d/vsftpd 中，是否设置错误，导致 Client 登录权限被取消了

2、已经成功连接，却显示"XXX file can't be open"的字样

- 最主要的原因在于 vsftpd.conf 当中设置了检查某个文件，但你却没有将这个文件设置起来，所以请检查 vsftpd.conf 里面所有设置的文件名，使用 touch 这个命令将该文件建立起来就行

3、Client 上已经连上 vsftpd 服务器，却无法使用某个账号登录

- vsftpd.conf 里面是否设置了使用 PAM 模块来检验账号，以及利用

`userlist_file` 来管理账号

- 检查`/etc/vsftpd/ftpusers` 以及`/etc/vsftpd/user_list` 文件内是否将该账号写入了

4、Client 无法上传文件

- 最可能的原因就是在 `vsftpd.conf` 里面忘记加上"write_enable=YES"这个设置
- 是否要上传的目录权限不对，用 `chmod` 或 `chown` 修改
- 是否 `anonymous` 的设置里面忘记加上了下面三个参数
`anon_other_write_enable=YES`
`anon_mkdir_write_enable=YES`
`anon_upload_enable=YES`
- 是否因为设置了 E-mail 阻挡机制，又将 E-mail address 写入该文件中
- 是否设置了不许 ASCII 格式传送，但 Client 端又以 ASCII 传送
- 检查`/var/log/message`，是否被 SELinux 阻挡了

5、还可以分析这两个文件：`/var/log/vsftpd.log` 与 `/var/log/messages`

- `/var/log/vsftpd.log` 默认不会出现，只有`/var/log/xferlog` 而已，若要加入 `/var/log/vsftpd.log` 的支持，在`/etc/vsftpd/vsftpd.conf` 增加如下两个设置
`dual_log_enable=YES`
`vsftpd_log_file=/var/log/vsftpd.log`

21.3. 客户端的图形接口 FTP 连接软件

21.3.1. Filezilla

1、<未完成>：略

21.3.2. 通过浏览器取得 FTP 连接

1、使用方法

`ftp://username@your_ip`

- 如果没有输入 `username@` 的字样时，系统默认会以匿名登录来处理这次连接

21.4. 让 vsftpd 增加 SSL 的加密功能

1、检查 vsftpd 有无支持 SSL 模块

- `ldd $(which vsftpd) | grep ssl`
- 若出现 `libssl.so` 字样，就说明有支持

2、专门建立给 vsftpd 使用的证书数据

- CentOS 给我们一个建立证书的地方，即`/etc/pki/tls/certs/`
- `cd /etc/pki/tls/certs`
- `make vsftpd.pem`

3、修改 `vsftpd.conf` 的配置文件，假定有实体，匿名账号

Chapter 22. 邮件服务器：Postfix

22.1. 邮件服务器的功能与工作原理

22.1.1. 电子邮件的功能与问题

- 1、夹带病毒的电子邮件问题
- 2、黑客通过邮件程序入侵
- 3、广告信与垃圾信等
- 4、主机被大量不明邮件塞爆
- 5、真实社会的讨厌事情
- 6、不实的邮件内容

22.1.2. Mail server 与 DNS 之间的关系

1、Mail server 与合法的主机名

- 现在已经没有人会使用 IP 来寄信了，通常收到 E-mail 都是使用"账号@主机名"的方式来处理
- 因此你的邮件服务器一定要有一个合法注册过的主机名才可以
- 只需要有一个合法主机名，在 DNS 的查询系统中你的主机名拥有一个 A 的标志，那么理论上 Mail server 就可以架设成功
- 但是由于目前因特网上的广告信，垃圾信与病毒信占用了太多带宽，导致整个网络社会消耗过多的成本在这些垃圾邮件上，为了杜绝垃圾邮件，目前的大型网络提供商(ISP)都会对不明来源的邮件加以限制，也就是说想要架设一台简单可以工作的 Mail server 越来越难了

2、DNS 的反解也很重要

- 对于一般的服务器来说，我们只要使用正解让客户端可以正确的找到我们服务器的 IP 即可架站
- 由于目前收信端的邮件服务器会针对邮件来源的 IP 进行反解，如果你的网络环境是由拨号取得而非固定的 IP 时，该种 IP 在 ISP 方面通常会主动的以 **xxx.dynamic.xxx** 之类的主机名来管理，但是这样的主机名会被主要的大型邮件服务器视为垃圾邮件
- 如果想要架设一台 Mail server 的话，请务必向你的上层 ISP 申请 IP 反解的对策，不要再使用默认的反解主机名

3、需要 DNS 的 MX 及 A 标识

- MX 代表的是 Mail eXchanger，当一封邮件要发送出去的时候，邮件主机会先分析那封信的目标主机的 DNS，先取得 MX 标识，然后以最优先 MX 主机为准将信发送出去
- 举例说明
 - 配置如下

```
xyz.com.vbird IN MX 10 mail.xyz.com.vbird
xyz.com.vbird IN MX 20 mail2.xyz.com.vbird
xyz.com.vbird IN A aaa.bbb.ccc.ddd
```
 - 当一封信要传给 user@xyz.com.vbird 时，由于 MX 标志最低者优先，所以该封信会先传送到 mail.xyz.com.vbird 那台主机
 - 如果 mail.xyz.com.vbird 由于种种原因，导致无法收下该封信时，该封信

将以次要 MX 主机来传送，那就是传送到 mail2.xyz.com.vbird 那台主机上

- 如果两台 MX 主机都无法负责，那么该封信会直接以 A 标志，直接传送到 aaa.bbb.ccc.ddd 那个 IP 上去，也就是 xyz.com.vbird 本身
- mail.xyz.com.vbird 与 mail2.xyz.com.vbird 必须要是可以帮 xyz.com.vbird 转发邮件的主机才行，也就是说，那两台主机通常是你公司的最上游的邮件主机，并不是随意填写的，那两台主机还需要针对你的 xyz.com.vbird 来设置邮件转发才行，否则你的信会被踢掉的
- 虽然不必自行设置 DNS 服务器，不过要申请一个 MX 标志才行，MX 标志一定要正确，否则邮件将可能会直接被 MX 服务器踢掉，但如果没有上层邮件服务器时，为了设置 MX，可以指定 MX 为自己，利用自己当 MX 服务器即可
- MX 的作用：一般来说，如果目标主机宕机，你的邮件通常会直接退给原发信人，但如果有 MX 主机，这台 MX 主机会先将该封信放在他的队列当中，等到你的目标主机重新提供邮件服务后，MX 主机会将你的邮件发送给目标主机

4、Email 的地址写法

- 账号@主机名

22.1.3. 邮件传输所需要的组件(MTA、MUA、MDA)及相关协议

1、要发送邮件，必须要先向某一台邮件服务器注册，以取得一个合法的电子邮件权限后，才能够发送邮件出去

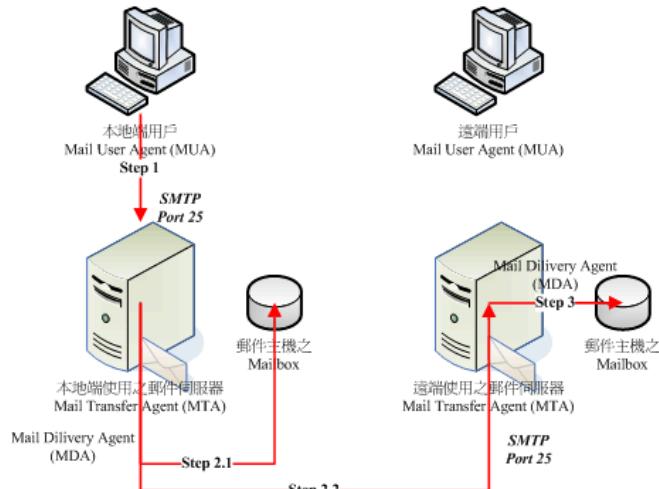


图 22-1 电子邮件传送过程示意图

2、MUA(Mail User Agent)

- MUA 就是邮件用户代理人的意思，除非你可以直接利用类似 Telnet 之类的软件登录邮件服务器来主动发出邮件，否则就需要通过 MUA 来帮你送信到邮件服务器
- 最常见的 MUA 如 Mozilla 推出的 Thunderbird(雷鸟)自由软件、Linux 桌面 KDE 常见的 Kmail 及 Windows 自带的 Outlook Express(OE)等
- MUA 主要的功能就是收邮件主机的电子邮件，以及提供用户浏览与编写邮件

3、MTA(Mail Transfer Agent)

- MUA 帮用户发送邮件到邮件主机上, 那这台邮件主机如果能够帮用户将这封邮件寄出去, 那它就是一台邮件发送主机(MTA)了
- MTA 就是邮件发送代理人的意思
- MTA 有如下功能
 - 接收邮件: 使用简单邮件传送协议(SMTP), MTA 主机最主要的功能就是: 将来自客户端或者其他 MTA 的邮件收下来
 - 转发邮件: 如果该封邮件的目的地并不是本身的用户, 且该封邮件的相关数据符合使用 MTA 的权利, 那么 MTA 就会将该封邮件再转发到下一台主机上, 这就是所谓的中继转发功能

4、MDA(Mail Delivery Agent)

- MDA 字面上的意思是"邮件传送代理人"
- MDA 是在 MTA 下面的一个小程序
- MDA 最主要的功能是: 分析由 MTA 所收到的邮件表头或内容等数据, 来决定这封邮件的去向
- 如果 MTA 收到的这封邮件目标是自己, 那么会将这封邮件转到用户邮箱(Mailbox)中, 如果不是, 就要转发出去了
- MDA 其他功能
 - 过滤垃圾邮件
 - 自动回复

5、Mailbox

- Mailbox 就是电子邮件邮箱, 就是某个账号专用的邮件收取文件
- Linux 系统默认的邮箱在/var/spool/mail/<用户账号>/中

6、MUA 将邮件送到对方的邮件邮箱(Mailbox)的步骤

- Step0: 取得某台 MTA 的权限, 即我们必须要向 MTA 注册一组可使用 E-mail 的账号与密码才行
- Step1: 用户在 MUA 上编写邮件后, 发送至 MTA 上, 邮件数据包括邮件标题和邮件内容
 - 编写完毕之后只要点击发送按钮, 该封邮件就会发送至你的 MTA 服务器上面了
 - 注意, 是你的 MTA 而不是对方的 MTA, 如果你确定可以使用该台 MTA, 那么你的这封邮件就会被放置到 MTA 的队列当中并等待发送出去了
- Step2.1: 如果该封邮件的目标是本地端 MTA 自己的账号, 那么会通过 MDA 将这封邮件送到 Mailbox 中去
- Step2.2: 如果该封邮件的目标为其他 MTA, 则开始中继转发(Relay)的流程, 此时 MTA 就会开始分析该封信是否具有合法的权限, 若具有权限时, 则我们的 MDA 会开始进行邮件转发, 也就是该封邮件会通过我们的 MTA 向下一台 MTA 的 SMTP(port25)发送出去, 如果该封邮件顺利地发送出去了, 那么该封邮件就会从队列当中删除
- Step3: 对方 MTA 服务器接受邮件, 如果一切顺利的话, 远程的 MTA 会收到我们 MTA 所发出的那封信, 并将该邮件放置到正确的用户邮箱当中, 等待用户登录来读取或下载
- 整个过程中, 邮件是由我们的 MTA 帮忙发送出去, 此时 MTA 提供的协议是简单邮件传输协议(Simple Mail Transfer Protocol,SMTP), 并且该封邮件最终是停留在对方主机的 MTA 上, 而不是你朋友的 MUA 上

22.1.4. 用户收信时服务器端所提供的相关协议：MRA

1、当用户想要收信时，当然也可以通过 MUA 直接来连接取得自己的邮件邮箱内的数据，整个过程如下图所示

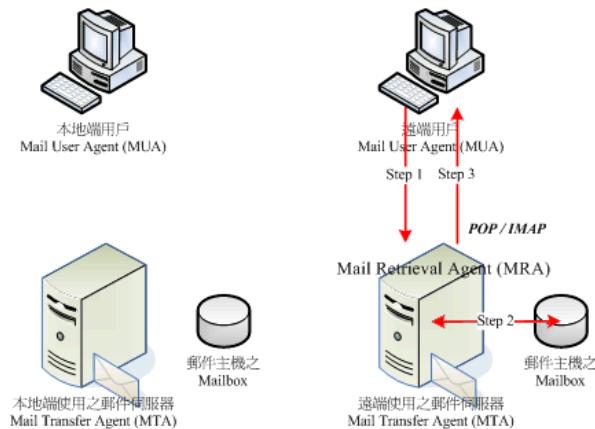


图 22-2 客户端通过 MRA 接受邮件的流程示意图

2、MRA(Mail Retrieval Agent)

- 用户可以通过 MRA 服务器提供的邮政服务协议(Post Office Protocol,POP)来接受自己的邮件，也可以通过 IMAP(Internet Message Access Protocol)协议将自己的邮件保留在邮件主机上面，并进一步建立邮件数据文件夹等高级工作
- 也就是说当客户端接受邮件时，使用的是 MRA 的 POP3、IMAP 等通信协议，并非 MTA 的 SMTP
- POP3 的收信方式
 - MUA 通过 POP3(Post Office Protocol version3)的协议连接到 MRA 的 port 110，并且输入账号与密码来取得正确的认证与授权
 - MRA 确认该用户账号 / 密码没有问题后，会前往该用户的 Mailbox(/var/spool/mail/用户账号)取得用户的邮件并发送到用户的 MUA 软件上
 - 当所有邮件传送完毕后，用户的 Mailbox 内的数据将会被删除
- IMAP(Internet Message Access Protocol)
 - 这个协议可以让你的 Mailbox 的数据存储到你主机上的用户主目录，亦即/home/账号/那个目录下，不但可以建立邮件数据文件夹，也可以针对邮件分类管理，而且在任何一个可连上网络的地方，只要登录主机，原本的邮件还是在的
- 要架设一台可以使用 MUA 进行收发邮件的 MTA、MRA 服务器，至少需要启动 SMTP 以及 POP3 这两个协议才行

3、POP3s、IMAPS 与 SMTP 的困扰

- SMTP、POP3、IMAP 等通信协议都是明文传输的，于是有了 POP3s、IMAPS 等通信协议出现
- 但是 SMTPs 协议虽然有，但是没人用
- POP3、IMAP 只与 MRA 及自己的用户有关，但是 MTA 必须与其他 MTA 沟通，若你使用了 SMTPs，那么全世界与你沟通的 MTA 都需要改为 SMTPs 通信协议才行，这个工程太浩大了

- 那么传输的数据一定是明文吗？虽然 MTA 无法加密，但是我们可以将邮件数据加密后再交由 MTA 传送即可

22.1.5. Relay 与认证机制的重要性

1、当需要 MTA 帮你将邮件转发到下一台 MTA 去时，这个操作就称为邮件中继转发(Relay)

2、如果所有人都可以通过一台 MTA 帮忙进行 Relay 时，这个情况就被称为 Open Relay 的操作，会造成以下问题

- 你主机所在的网络正常使用的连接速度将会变慢，因为网络带宽都被广告、垃圾耗光了
- 你的主机可能由于大量发送邮件导致主机资源被耗尽，容易产生不明原因宕机之类的问题
- 你的 MTA 将会被因特网社会定义为"黑名单"，从此很多正常的邮件就无法收发
- 你 MTA 所在的这个 IP 会被上层 ISP 所封锁，知道你解决这个 Open Relay 的问题
- 某些用户将会对你的能力产生质疑，对你公司或者是个人失去信心，甚至可能流失客源
- 如果你的 MTA 被利用来发黑客邮件，你是找不到原发信人的，所以你这台 MTA 将会被追踪为最终站

3、因此，目前所有的 distributions 都一样，几乎都将 MTA 默认启动为仅监听内部循环接口(lo)而已，而且也将 Open Relay 的功能取消了

4、因此，要想使用 MTA 的 Relay 功能，必须取得合法使用该 MTA 的权限，也就是说，设置谁可以使用 Relay 的功能就是我们管理员的任务，通常设置 Relay 的方法有以下几种

- 规定某一个特定客户端的 IP 或网段，例如规定内部 LAN 的 192.168.1.0/24 可以使用 Relay
- 若客户端的 IP 不固定，可利用认证机制来处理
- 将 MUA 搭建在 MTA 上面，例如 OpenWebMail 之类的 Web 接口的 MUA 功能

22.1.6. 电子邮件的数据内容

1、E-mail 会有几个重要的信息，包括

- 这封信来自哪个 MTA
- 这封信是由谁发送出来的
- 这封信要送给谁
- 这封信主题为何
- 这封信内容

22.2. MTA 服务器：Postfix 基础设定

1、可实现 MTA 的服务器软件非常多，例如 CentOS 默认提供了 SendMail，以及近期很热门的 Postfix

- 由于 SendMail 的配置文件太难懂，以及早期的程序漏洞问题导致的安全

- 性缺失，加上 SendMail 所有功能都总和在 /usr/sbin/sendmail 这个程序中，导致程序太大，可能会有效率方面的问题
- 新版的 CentOS 已经将默认的 Mail server 调整为 Postfix 了

22.2.1. Postfix 开发

1、Postfix 主要是针对想要完全兼容与 SendMail 所设计出来的一款内在部分完全新颖的一个邮件服务器软件

22.2.2. 所需要的软件与软件结构

1、由于 CentOS 默认提供 Postfix，因此无需调整什么(???好像没有哦)

- yum install postfix

2、主要的配置文件都在 /etc/postfix/ 当中

- **/etc/postfix/main.cf**: Postfix 主要的配置文件

- 几乎所有的设置参数都在这个文件内规范的，这个文件默认就是一个完整的说明文件了
- 修改过该文件需要重启 postfix

- **/etc/postfix/master.cf**: 规定了 Postfix 每个程序的工作参数，也是很重要的一个配置文件，不过这个文件默认已经很好了，通常不需要修改它

- **/etc/postfix/access**(利用 postmap 处理): 可以设置开放 Relay 或拒绝连接的来源或目标地址等信息的外部配置文件

- 这个文件要生效的话，需要在 /etc/postfix/main.cf 启动才行
- 设置完毕后需要以 postmap 来处理称为数据库文件

- **/etc/aliases**(利用 postalias 或 newaliases 均可): 作为邮件别名的用途，也可作为邮件组的设置

- **/usr/sbin/postconf**(查阅 Postfix 的设置数据): 这个命令可以列出当前你的 Postfix 的详细设置数据

- 包括系统默认设置，数据量庞大
- 如果仅要列出非默认的设置数据，则可以使用 "postconf -n"

- **/usr/sbin/postfix(主要的 deamon 命令)**:

- postfix check <== 检查 Postfix 相关的文件、权限是否正确
- postfix start <== 开始 Postfix 执行
- postfix stop <== 关闭 Postfix
- postfix flush <== 强制将目前正在邮件队列的邮件寄出
- postfix reload <== 重新读入配置文件，也就是 /etc/postfix/main.cf

- **/usr/sbin/postalias**: 设置别名数据库的命令

- 由于 MTA 读取数据库格式的文件效率较好，所以我们都会将 ASCII 格式的文件重建为数据库
- 在 Postfix 中，这个命令主要用于转换 /etc/aliases 成为 /etc/aliases.db
postalias hash:/etc/aliases

- **/usr/sbin/postcat**: 主要用于检查放在 queue(队列) 中的邮件内容

- 由于队列当中的邮件内容是给 MTA 看的，所以格式并不是一般我们能看得懂的文字数据，这个时候需要用 postcat 才可以看出该邮件的内容
- 在 /var/spool/postfix 内有相当多的目录，假设内有一个文件名为 /deferred/abcfile，那可以用如下方式来查询该文件内容

- postcat /var/spool/postfix/deferred/abcfile
- /usr/sbin/postmap: 这个命令的用法与 postalias 类似, 不过它主要用在转换 access 文件的数据库
postmap hash:/etc/postfix/access
- /usr/sbin/postqueue
 - 类似 mailq 的输出结果

22.2.3. 一个邮件服务器的设定案例

1、要搭建一台可以连上 Internet 的邮件服务器时, 你必须要已经取得合法的 A 与 MX 主机, 而且最好反解也已经向你的 ISP 申请修改设置了

2、主要设置内容

- 邮件服务器主要名称为: www.centos.liuye
- 邮件服务器尚有别名为 linux.centos.liuye 以及 ftp.centos.liuye 也可以收发邮件
- 此邮件服务区已有 MX 设置, 直接指向自己(www.centos.liuye)

22.2.4. 让 Postfix 可监听 Internet 来收发邮件

1、/etc/postfix/main.cf 配置文件的注意事项

- #符号是注释的意思
- 所有设置值以类似变量设置方法来处理, 例如


```
myhostname = www.centos.liuye
```

 - 等号两边要给予空格符
 - 第一个字符不能是空白
- 可以使用\$来延伸使用变量设置
- 如果该变量支持两个以上的数据, 则使用空格符来分隔, 不过建议使用逗号加空格符来处理


```
mydestination=$myhostname, $mydomain, linux.centos.liuye
```
- 可用多行来表示同一个设置值, 只要在第一行最后有逗号, 且第二行开头为空格符, 即可将数据延伸到第二行继续书写
- 如哦重复设置某一项目, 则以较晚出现的设置值为准

2、必须要设置的数据

- myhostname: 设置主机名, 需要使用 FQDN
- myorigin: 发信时所显示的"发信源主机"项目
 - 这个项目在设置"邮件头上面的 mail from 的那个地址"
 - 即代表本 MTA 传出去的邮件将以此设置值为准
 - 如果在本机寄信时忘记加上 Mail from 字样的话, 那就以本设置值为准
 - 默认这个项目以\$myhostname 为主
- inet_interfaces: 设置 Postfix 的监听接口(极重要)
 - 在默认情况下你的 Postfix 只会监听本机接口的 lo
 - 常见的设置方法为 inet_interfaces=all
- inet_protocols: 设置 Postfix 监听 IP 协议
 - 默认 CentOS 的 Postfix 会同时监听 IPv4、IPv6 两个版本的 IP, 如果环境仅有 IPv4, 那可以直接指定 inet_protocols=ipv4
- mydestination: 设置"能够收信的主机名"(极重要)

- 这个项目非常重要，因为我们的主机有非常多的名字，那么对方填写的 mail to 到底写哪个主机名字我们才能将该邮件收下，就是在这里规范的
- **如果你的 DNS 里的设置有 MX 标志的话，那么请将 MX 指向的那个主机名字一定要卸载这个 mydestination 里，否则很容易出现错误信息，这也是最常见的错误**
- mynetworks_style: 设置"信任网络"的一项指标
 - 这个设置值在规定与主机在同一个网络的可信任客户端
 - 一般来说，mynetworks 会取代这个设置值，因此不设置也没有关系
 - 如果要设置的话，最好设置成为 host 即可，即仅信任这部 MTA 主机而已
- mynetworks: 规定信任的客户端(极重要)
 - 你的 MTA 能不能帮忙进行 Relay 与这个设置值有很大关系
 - 当要开放本机与内部网络的 IP 时，就可以这样设置
mynetworks=127.0.0.0/8, 192.168.100.0/24
 - 如果想要以/etc/postfix/access 这个文件来控制 relay 的用户时，可以改写为以下，然后再建立 access 重整数据库后，就能设置 Relay 的用户了
mynetworks=127.0.0.0/8, 192.168.100.0/24, hash:/etc/postfix/access
- relay_domains: 规范可以帮忙 relay 的下一台 MTA 主机地址
 - 相对于 mynetworks 是针对"信任的客户端"而设置的，这个 relay_domains 可以视为"针对下游 MTA 服务器"而设置的
 - 在默认情况下，这个设置值是\$mydestination
 - Postfix 默认并不会转发 MX 主机的邮件
- alias_maps: 设置邮件别名

3、具体设置参考 P767

- postmap hash:/etc/postfix/access
- postalias hash:/etc/aliases
- postfix check
- systemctl restart postfix.service
- netstat -tunlp | grep :25

22.2.5. 邮件发送流程与收信、Relay 等重要概念

1、一般来说，一封邮件发送出去会经过的流程如下

- 1) 发信端与首先断两台主机先经过一个握手(ehlo)的阶段，此时发信端被记录为发信来源(而不是 Mail from)，通过握手后就可以进行邮件标题(header)的传送
- 2) 此时收信端主机会分析标题的信息，若邮件已"Mail to:主机名"为收信端主机，且该名称符合 mydestination 的设置，则该邮件会开始被接受到队列，并进一步送到 Mailbox 当中，若不符合 mydestination 的设置，则终止连接且不会进行邮件内容的传送
- 3) 若"Mail to:主机名"不是收信端本身，则开始进行中继转发(Relay)的分析
- 4) 转发过程首先分析该邮件的来源是否符合信任的客户端(这个客户端为步骤 1 所记录的发信端主机)，**也就是来源是否符合 mynetworks 的设置值**，若符合则开始接收邮件至队列中，并等待 MDA 将邮件再转发出去，若不符合则继续下一步

- 5) 分析邮件来源或目标是否符合 `relay_domain` 的设置，若符合则邮件将被接收至队列，并等待 MDA 将邮件再转发出去
- 6) 若这封邮件的标题数据都不符合上述的规范，则终止连接，并不会接收邮件的内容数据

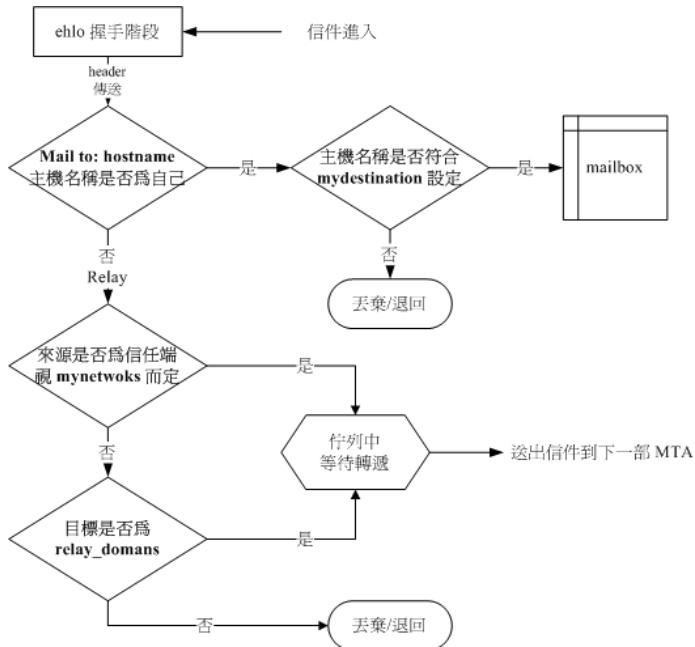


图 22-3 在本机 MTA 当中的邮件分析过程

- 2、也就是说，标题分析通过后，你的邮件内容才会开始上传到主机的队列，然后通过 MDA 来处理邮件的流向，而不是将邮件完整地传送到主机后才开始分析
- 3、收信方面

- 发信端必须符合\$inet_interfaces 的设置
- 邮件标题的收件人主机名必须符合\$mydestination 的设置，或者收件主机需要符合\$virtual_map(与虚拟主机有关)的设置

4、转发方面

- 发信端必须符合\$inet_interfaces 的设置
- 发信端来源必须为\$mynetworks 的设置，发信端来源或邮件标题的收件人主机名符合\$relay_domains 的设置内容

22.2.6. 设定邮件主机权限与过滤机制：/etc/postfix/access

- 1、基本上，指定了 Postfix 的 \$mynetworks 的信任来源就能够让用户 Relay 了，我们还可以利用 access 这个文件来额外管理我们的邮件过滤(需要参考 P767 进行设置才行)，基本的语法如下

| 规范的范围或规则 | Postfix 的操作 |
|--|-------------|
| IP/部分 IP/主机名/Email 等 | OK/REJECT |
| ➤ 这个文件设置最大的好处就是，你不必重新启动 Postfix，只要将数据库建立好，立刻就生效了 | |

22.2.7. 设定邮件别名：/etc/aliases、~/.forward

- 1、很多系统账号，例如 named、apache、mysql 等，这些账号执行的程序若有信息发生时，会将该信息以 E-mail 的方式传给 root，因为系统账号并没有密码可登

陆，自然就无法接受邮件了，所以若有邮件就给系统管理员

- MTA 如何知道这些邮件要传给 root，这就需要 aliases 这个邮件别名配置文件来处理了

2、邮件别名配置文件：/etc/aliases

- etc/aliases 文件，左边是别名，右边是实际存在的用户账号或者是 E-mail address
- 通过这个设置，我们可以将所有系统账号所属的邮件全部丢给 root
- 例子
 - vim /etc/aliases
 - dermintsa: dmtsa
 - postalias hash:/etc/aliases

3、/etc/aliases 实际应用一：让一般账号可接受 root 的邮件

- vim /etc/aliases
- root: root,dmtsa! <==推荐用这种
- root: dmtsa! <==会导致 root 收不到邮件了
- postalias hash:/etc/aliases

4、/etc/aliases 实际应用二：配置组寄信功能

- vim /etc/aliases
- student2011: std001,std002,std003...
- postalias hash:/etc/aliases
- 还可以填写外部主机的 E-mail: 用户名@主机名

5、个人化的邮件转递：~/.forward

- 虽然/etc/aliases 可以帮我们实现邮件别名的设置，不过/etc/aliases 是只有 root 才能修改的文件权限
- 一般用户如果也想要进行邮件转发时，可以通过自己用户主目录下的.forward 这个文件
- 这个文件内容是一行一个账号(或者 E-mail)，而且权限方面非常重要，改为 644，不让 Group 以及 Other 进行写入

22.2.8. 查看邮件队列信息：postqueue、mailq

1、在队列中等待送出的邮件

- 如果该信在五分钟之内无法寄出，通常系统会发出一封警告信给原发件人，告知该封邮件尚无法被寄送出去，不过系统仍然会持续尝试寄出该封邮件
- 如果在 4 小时后仍无法寄出，系统会再次发出警告信给原发件人
- 如果持续进行 5 天都无法将邮件送出，那么该封邮件就会退回给原发件人

2、让邮件卡在队列中(用于查看队列信息之用)

- systemctl stop postfix.service
- echo "test" | mail -s "testing queue" root
- postqueue -p
- 输出的信息解释
 - QueueID: 表示此封邮件队列的代表号(ID)，这个号码是给 MTA 看的
 - Size: 这封邮件有多大容量(bytes)
 - Arrival Time: 这封信什么时候进入队列的，并且可能会说明无法立即发送出去的原因

- Sender/Recipient: 送信人与收件人的电子邮件

3、可以利用 postcat 读出原邮件的内容

- cd /var/spool/postfix/maildrop <==队列中的邮件是放在此处的
- postcat <Queue ID>

4、重新将队列当中的邮件寄出去

- systemctl restart postfix.service
- postfix flush

22.2.9. 防火墙设置

1、整个 MTA 主要是通过 SMTP(port 25)进行邮件发送的任务

22.3. MRA 服务器: dovecot 设定

1、收信所要用到的通信协议: POP3 以及 IMAP

2、CentOS 使用的是 dovecot 这个软件来实现 MRA 的相关通信协议的

22.3.1. 基础的 POP3/IMAP 设定

1、启动 POP3/IMAP

- yum install dovecot
- 该软件的配置文件仅有/etc/dovecot/dovecot.conf
- vim /etc/dovecot/dovecot.conf
 - protocols = imap pop3 <==大约第 25 行
- vim /etc/dovecot/conf.d/10-ssl.conf
 - ssl = no <==大约第六行

22.3.2. 加密的 POP3s/IMAPs 设定

1、与 Apache 相似，都是通过 openssl 这个软件提供 SSL 加密机制来进行数据的加密传输

2、默认情况下，CentOS 已经提供了 SSL 证书范例文件使用了，如果不想使用默认证书，可以按照如下方式建立一个

- 生成证书
 - cd /etc/pki/tls/certs/
 - make liuyedovecot.pem <==make [证书文件名.pem]
- 将 pem 文件放到 dovecot 并更改 SELinux 的字段类别
 - mv liuyedovecot.pem /etc/dovecot
 - restorecon -Rv /etc/dovecot
- 开始处理 dovecot.conf
 - vim /etc/dovecot/conf.d/10-auth.conf
 - disable_plaintext_auth = yes <==取消注释
 - vim /etc/dovecot/conf.d/10-ssl.conf
 - ssl = required <==大约第六行
 - ssl_cert = </etc/dovecot/liuyedovecot.pem <==大约 12、13 行
 - ssl_key = </etc/dovecot/liuyedovecot.pem
 - vim /etc/dovecot/conf.d/10-master.conf

```

inet_listener imap {
    port = 0           <==大约 15 行
}

inet_listener pop3 {
    port = 0           <==大约 36 行
}
● vim /etc/dovecot/conf.d/10-mail.conf
mail_location = mbox:~/mail:INBOX=/var/mail/%u   <==大约 30 行
➤ systemctl restart dovecot

```

22.3.3. 防火墙设置

22.4. MUA 软件：客户端的收发邮件软件

22.4.1. Linux mail

1、在 Unix like 的操作系统当中都会存有一个可进行收发邮件的软件，那就是 **mail** 这个命令，这个命令是由 **mail** 这个软件提供的，因此需要先安装这个软件

- 由于 **mail** 是 Linux 系统的功能，所以即使 **port 25(SMTP)** 没有启动，他还是可以使用的，只是该封邮件就只会被放到队列，而无法寄出去

2、用 **mail** 直接编辑文字邮件与发送邮件

- **mail** 的用法很简单，就是利用"**mail [E-mail address]**"的方式来将邮件发送出去，[E-mail address]可以是对外的邮件地址，也可以是本机的账号，如果是本机账号的话，只需要账号名即可
- 对外寄信的时候，邮件默认的 **Mail from** 就会填写 **main.cf** 内的 **myorigin** 变数的主机名
- 例子

mail liyehcf@163.com

Subject : Just test <==填写标题

This is a test email!. <==填写邮件内容

. <==该行只有小数点，代表输入结束的意思

3、利用已经处理完毕的"纯文本文档"发送邮件

- **mail -s 'My bashrc' liyehcf@163.com < ~/bashrc**
- 不知道什么原因，收不到，执行以下 **postfix flush**

4、开始查阅接受的邮件

- **mail**
 - 该命令会主动获取用户在 **/var/spool/mail** 下面的邮件邮箱
- 可以使用的命令
 - 读信：直接按 **Enter** 键或输入数字后按 **Enter** 键
 - 显示标题： "h" 或 "h[数字]"
 - 回复邮件：直接输入 **R**
 - 删除邮件：输入 "d[数字]"，例如 **d2, d10-50**
 - 存储邮件到文件：输入 "s[数字] [文件名]"
 - 离开 **mail**： **q** 或 **x**

- 输入 **x** 可以在不更动 Mailbox 的情况下离开 mail 程序，不管你刚才有没有使用 **d** 删除数据
 - 使用 **q** 才会将删除的数据移除
- 读取转存的邮件
- **mail -f [文件名]**
- 5、以添加附件的方式寄信
- 需要使用到 **uuencode** 这个命令，在 CentOS 中这个命令属于 **sharutils**
 - **yum install sharutils**
 - 使用方法
 - **uuencode [实际文件] [邮件中的文件名] | mail -s '标题' email**
 - **uuencode /etc/hosts myhosts | mail -s 'test encode' liuye**
 - **mail**，然后利用 **s[数字] [文件名]** 将邮件存下来
 - **uudecode [加密文件名] -o [输出文件名]**

22.4.2. Linux mutt

- 1、mutt 除了可以仿真 mail 这个命令之外，它还能够通过 POP3/IMAP 之类的协议去读取外部的邮件
- 2、直接以<mutt>进行发送邮件的操作：含快速添加附件文件
- **mutt [-a 附加文件] [-i 内文件] [-b 秘密副本] [-c 一般副本] **
[-s 邮件标题] E-mail 地址
 - **-a** 附加文件：后面就是想要传送的文件，而不是邮件内容
 - **-i** 内文件：邮件内文部分，先编写称为文件而已
 - **-b** 秘密副本：原收件人不知道这封信还会寄给后面那个秘密副本收件人
 - **-c** 一般副本：原收件人会看到这封信还有传给哪位收件人
 - **-s** 邮件标题：这封信的标题
 - **E-mail 地址：**就是原收件人的 E-mail
 - 例子
 - **mutt -s '一封测试信' liyehcf@163.com**
 - 会进入 vi 编辑界面，编辑完邮件正文后，**:wq**，然后根据屏幕上方的提示来进行进一步操作
 - **mutt -s 'hosts' -i /etc/hosts liyehcf@163.com**
 - 添加附件
 - **mutt -s '附件' -a /usr/hosts -- liyehcf@163.com**
 - **"-a 文件名"必须出现在命令的最后面**
 - **文件名与 E-mail 地址之间需要加上两个连续减号(--)才行**

3、以 mutt 来读不通通信协议的邮箱

- 与 mail 相比，mutt 可以直接通过网络的 POP3、IMAP 等通信协议来读信，是相当优秀功能
- **mutt [-f 邮箱位置]**
 - **mutt -f imaps://服务器的 IP**
- 没太搞懂

22.4.3. 好用的跨平台(Windows/Linux)软件：Thunderbird

- 1、自由软件最大的好处之一就是该软件大多可以进行移植，也就是在任何操作

系统上面几乎都能执行该软件。MUA 也有自由软件，那就是 Mozilla 基金会推出的 ThunderBird

22.5. 邮件服务器的高级设定

1、目前，邮件攻击的主要问题已经不是病毒与木马了，大多数的垃圾邮件多是以钓鱼以及色情广告

- 网络钓鱼的问题在于用户的好奇心以及糟糕的操作习惯

2、下面主要针对 Postfix 的邮件收件过滤处理，以及重新发送的 Relay 过程进行介绍，这两个过程在 Postfix 的设置中，主要有以下几个重要的项目管理

- `smtpd_recipient_restrictions`: recipient 是收件人的意思，这个设置值主要负责管理由本级所收下的邮件的功能，因此大部分的设置都是在进行邮件过滤以及是否为可信任邮件，来源可以是 MTA 或 MUA
- `smtpd_client_restrictions`: Client 是客户端的意思，因此主要负责管理客户端的来源是否可信任。可以将非 Mail server 来信拒绝掉。来源指 MUA
- `smtpd_sender_restrictions`: sender 是发件人的意思，可以针对邮件来源(对方邮件服务器)进行来源分析过滤的操作。来源理论上就是 MTA

22.5.1. 邮件过滤一：用 postgrey 进行非正规 Mail server 的垃圾邮件过滤

1、僵尸计算机发送邮件的特色

- 它只会尝试传送该封邮件一次，无论是否成功，该封信就算发出去了，故该邮件将被从队列中删除

2、根据合法与非法邮件服务器工作流程而发展处一套曙光(postgrey)软件

- postgrey 主要的功能是记录发信来源而已
 - 若发信来源同一封信第一次寄来时，postgrey 默认会过滤他，并且将来源地址记录下来，在约 5 分钟后，该邮件又传来一次时，则该邮件会被收下来
 - 如此一来便可以杜绝非法邮件服务器单词发送的问题
- 对于确定合法的主机则可以开放所谓的"白名单"来优先通过而不过滤
 - 确认发信来源是否在白名单中，若是则予以通过
 - 确认收信者是否在白名单中，若是则予以通过
 - 确认这封信是否已经被记录下来，放行的依据是
 - 若无此邮件的记录，则将发信地址记录下来，并将邮件退回
 - 若有此邮件的记录，但是记录的事件未超过指定时间(默认 5 分钟)，则依旧退回邮件
 - 若有邮件的记录，且记录时间已超过指定的时间，则予以通过

3、由于 postgrey 有记录能力，因此数据库系统不可避免，又由于 postgrey 是由 perl 写成的，可能需要加入很多相关的 perl 模块才行，需要的软件如下

- Berkeley DB: 包括 db4,db4-utils,db4-devel
- Perl
- Perl 模块

4、<未完成>：不知道咋装

22.5.2. 邮件过滤二：关于黑名单的过滤机制

- 1、一般来说，只要是 Open Relay 的邮件 MTA 都会被列入黑名单中，因此要特别注意 Open Relay 的问题
- 2、我们可以自行前往该网站将有问题的主机列表加入自己的邮件主机过滤机制当中，不过由于因特网上已经提供黑名单数据库了，我们就可以利用这个数据库来阻挡

➤ 在决定是否进行 Relay 之前，先要求 Postfix 前往追踪黑名单的数据库，若目标的 IP 或主机名在黑名单的医院，则我们就将该邮件拒绝

- 3、检查你的邮件服务器是否在黑名单中

➤ 是否已在黑名单数据库中：直接登录 <http://cbl.abuseat.org/lookup.cgi>
➤ 是否具有 Open Relay：直接登录 <http://rs.edu.tw/tanet/spam.html>
➤ 如何删除：关闭 Open Relay 功能，改善过后联系管理员

22.5.3. 邮件过滤三：基础的邮件过滤机制

- 1、在整封信的传送流程中，客户端若通过主机的重重限制后，最终应该可以到达邮件队列当中。而由队列当中要送出去或者是直接送到 Mailbox 就需要通过 MDA 的处理

➤ MDA 可以加挂很多机制，尤其是可以过滤某些特殊字眼的广告邮件或者病毒邮件
➤ MDA 可以通过分析整封邮件的内容(包括标题以及内容)来摘取有问题的关键词，然后决定这封信的命运

- 2、Postfix 已经有内建可以分析标题或者是内容的过滤机制了，即/etc/postfix/目录下的 header_checks 以及 body_checks 这两个文件

➤ 默认情况下这两个文件不会被 Postfix 使用，需要在/etc/postfix/main.cf 中启用它们
➤ vim /etc/postfix/main.cf

- header_checks = regexp:/etc/postfix/header_checks
- body_checks = regexp:/etc/postfix/body_checks
- touch /etc/postfix/header_checks
- touch /etc/postfix/body_checks

- 3、自行处理 header_checks 以及 body_checks 的规则设置

➤ 语法

- #代表注释
- 默认不区分大小写
- 规则设置方法：
/规则/ 操作 显示在登录文件里面的信息
/^Subject:.*A funny game/ DISCARD dropheader deny

➤ 操作

- REJECT：将该封邮件退回给发件人
- WARN：将邮件收下来
- DISCARD：将该封邮件丢弃，并不给予原发件人回应

➤ 设置完毕后语法检查
postmap -q - regexp:/etc/postfix/body_checks < /etc/postfix/body_checks

- 注意-与 regexp 中间的空格

22.5.4. 非信任来源的 Relay：开放 SMTP 身份认证

1、由 MUA 通过 MTA 来发送邮件时(具有 Relay 操作时), 理论上 MTA 必须要开放信任用户来源才行, 这就是我们必须要在 main.cf 里设置 smtp_recipient_restrictions 这个设置项目(my networks)的原因

2、但如果用户取得的 IP 非固定的话, 那么这时候就需要 SMTP 认证了

- 让你在想要使用 MTA 的 port 25(SMTP 协议)时, 需要输入账号密码才行
- 既然有了认证的功能, 就不用设置 MTA 的信任用户项目

3、如何让 SMTP 支持身份认证? CentOS 已经提供内建的认证模块, 即 Cyrus SASL 这个软件

- 在 SMTP 认证方面, Cyrus 主要提供了 saslauthd 这个服务来进行账号密码的比对操作
- 当有任何人想要进行邮件转发功能时, Postfix 会联系 saslauthd 请其代为检查账号密码, 若比对通过则允许客户端开始转发邮件

4、配置步骤

- 安装 cyrus-sasl、cyrus-sasl-plain、cyrus-sasl-md5 等软件
- 启动 saslauthd 这个服务
- 设置 main.cf 让 postfix 可以与 saslauthd 联系
- 客户端必须要在寄信是设置邮件主机认证功能

5、启动 saslauthd 服务: 进行 SMTP 明文身份验证功能

- 如果想要直接使用 Linux 系统上面的用户信息, 也就是 /etc/passwd、/etc/shadow 所记载的账号密码相关信息时, 可以使用 saslauthd 提供的 shadow 这个机制, 当然也能使用 pam
- 了解 saslauthd 支持哪些密码管理机制
saslauthd -v
- 在 saslauthd 配置文件中, 选定 pam 的验证机制
vim /etc/sysconfig/saslauthd
MECH=pam
- systemctl restart saslauthd.service
- 需要告知 Cyrus 使用提供 SMTP 服务的程序为 saslauthd 才行
vim /etc/sas12/smtpd.conf
log_level: 3 <== 登录文件信息的等级
pwcheck_method: saslauthd <== 选择什么服务来负责密码的比对
mech_list: plain login <== 支持的机制

6、更改 main.cf 设置项: 让 postfix 支持 SMTP 身份验证

- vim /etc/postfix/main.cf
smtpd_sasl_auth_enable = yes
smtpd_sasl_security_options = noanonymous
broken_sasl_auth_clients = yes
<未完成>: 心累

7、在客户端启动支持 SMTP 身份验证的功能 P798

22.5.5. 非固定 IP 邮件服务器的福音: relayhost

1、当你的 MTA 要发邮件给目标 MTA 时, 如果直接传给目标 MTA, 由于你的 IP

可能是非固定的，因此对方 MTA 恐怕会把你当成是垃圾来源

2、我们可以通过 ISP 进行转发，当我们要传给目标 MTA 时

- 先将邮件交给你的 ISP，因为你是 ISP 的客户，通常来信都会被 ISP 接受，因此这个时候这封信就会被你的 ISP 给 Relay 出去
- 被 ISP 锁 Relay 的邮件到目标 MTA 时，对方会判断是来自那台 ISP 的 MTA，当然是合法的 Mail server，于是该封邮件就被收下了

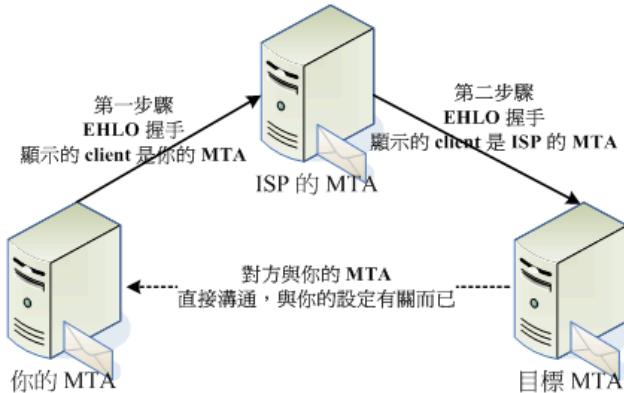


图 22-4 relayhost: 利用 ISP 的 MTA 进行邮件传递

3、以上述方法架设 MTA 的注意点

- 需要有一个合法的主机名，若要省钱，可以使用 DDNS 来处理
- 你上层的 ISP 所提供的 MTA 必须要有提供你所在 IP 的 Relay 权限

4、设置方法

- `vim /etc/postfix/main.cf`
- `relayhost = [<ISP 主机名>]`

22.5.6. 其他设置小技巧

1、单封邮件与单个邮箱的大小限制

- 默认情况下，Postfix 可接受的单封邮件最大容量为 10MB
- 若要修改，则需要修改/etc/postfix/main.cf 中的 message_size_limit 设置值（单位 bytes）即可，更改完后执行 postfix reload

2、之前管理邮箱容量用的是 quota，现在可以直接配置

- 修改/etc/postfix/main.cf 中的 mailbox_size_limit 设置值（单位 bytes）即可

3、发件备份：SMTP 自动转发一份到备份文件夹

- 收件备份可以用/etc/aliases 来处理
- 修改/etc/postfix/main.cf 的 always_bcc 设置值即可

4、配置文件的权限问题：权限错误会不能启动 Postfix

- /etc/aliases 仅能由系统信任的账号来修改，通常为 644
- Mail server 读取的数据库，多半在/etc/mail/或/etc/postfix/下面的*.db 文件，仅能由系统信任的用户读取，其他一概不能读取，通常为 640
- 系统的队列目录(/var/spool/mqueue 或/var/spool/postfix)仅允许系统读取，通常为 700
- 确定~/.forward 这个文件的权限不能设置成为任何人都能查阅，否则你的 E-mail 数据可能会被窃取

5、备份资料：与 mail 有关的目录有哪些

- /etc/passwd /etc/shadow /etc/group 等于账号有关的资料
- /etc/mail /etc/postfix 下面的所有文件数据
- /etc/aliases 等 MTA 相关文件
- /home 下面的所有用户数据
- /var/spool/mail 下面的文件与 /var/spool/postfix 邮件队列文件

6、错误检查：查出不能启动 Postfix 的问题流程

- 关于硬件配置
- 关于网络参数的问题
- 关于服务的问题
- 关于防火墙的问题
- 关于配置文件的问题
- 其他文件的设置问题
 - 只有某个 domain 可以收信，其他同一主机的 domian 无法收信，需要检查 \$mydestination 的设置值
 - 发现邮件被挡下来了，而且老是显示 reject 的字样，可能被 access 挡住了
 - 如果发现邮件队列中存在很多邮件，可能 DNS 死掉了，检查 /etc/resolv.conf
- 其他可能的问题：最有可能就是认证问题
- 还是不知道问题的解决方案：/var/log/message

各种问题

1、Windows 可以 ping 通 Linux 主机，但是 Linux 主机无法 ping 通 Windows

- 控制面板--系统和安全---Windows 防火墙---左边的"高级设置"
- 入站规则，中间明细找到如下图的设置项
- 右键 ICMPv4-In 点属性---作用域---添加需要放行的网段或者主机 IP

| | | | |
|--|----------|----|---|
| 文件和打印机共享(回显请求 - ICMPv4-In) | 文件和打印机共享 | 专用 | 否 |
| 文件和打印机共享(回显请求 - ICMPv4-In) | 文件和打印机共享 | 域 | 否 |
| <input checked="" type="checkbox"/> 文件和打印机共享(回显请求 - ICMPv4-In) | 文件和打印机共享 | 公用 | 是 |
| 文件和打印机共享(回显请求 - ICMPv6-In) | 文件和打印机共享 | 专用 | 否 |
| 文件和打印机共享(回显请求 - ICMPv6-In) | 文件和打印机共享 | 域 | 否 |
| <input checked="" type="checkbox"/> 文件和打印机共享(回显请求 - ICMPv6-In) | 文件和打印机共享 | 公用 | 是 |

2、Windows 上一台 Linux 虚拟机主机开启了 dhcp 服务，并且关闭了 VM Ware 自带的 dhcp 服务

- Windows 上的默认路由变成了 Linux 虚拟机主机的默认网关
- 于是 Windows 无法 ping 通其他的路由了
- 成了一个死循环，因为 Linux 虚拟机的默认路由是接到 Windows 上的，而 Windows 的默认路由却是接的 Linux 虚拟机

3、Windows 上开启了一台 Linux 虚拟机，虚拟机有两个网卡接口分别在两个网段

- Windows 可以 ping 通处于同一网段的 Linux 虚拟机 IP

- Windows 无法 ping 通处于不同网段的 Linux 虚拟机 IP
- Linux 已经开启数据包转发功能

杂项

1、192.168.0.1/24

- 代表网络时，24 代表子网掩码中 1 的个数
- 代表 IP 时，24 可能是 port num