LPI 考试准备: 系统安全性

中级管理（LPIC-2）主题 212

David Mertz (mertz@gnosis.cx)
开发人员
Gnosis Software, Inc.

2006 年 7 月 20 日

本文是由 7 篇文章 组成的介绍 Linux® 上中级网络管理的系列教程中的第 6 篇。在本教程中，David Mertz 将继续带领大家准备 Linux Professional Institute Intermediate Level Administration（LPIC-2）Exam 202 的考试。本教程将根据需要从安全敏感网络服务器的角度来简要介绍一些与 Linux 有关的主题，包括路由、防火墙、NAT 转换以及相关工具的常见问题。本文将解决为 FTP 和 SSH 设置安全策略时存在的问题，回顾如何使用 tcpd、hosts.allow 以及相关的文件来实现通用的访问控制，还将介绍一些基本的安全监视工具，以及在什么地方可以找到安全资源。

查看本系列更多内容

开始之前

了解这些教程可以让您学习到哪些知识，以及如何从这些教程中学习最多的知识。

关于本系列教程

Linux Professional Institute（LPI）对 Linux 系统管理员的认证分为两级：初级（也称为 "认证级别 1"）和中级（也称为 “认证级别 2”）。要获得认证级别 1，则必须通过 101 和 102 的考试；要获得认证级别 2，则必须通过 201 和 202 的考试。

developerWorks 上给出了一些可以帮助您准备这些考试的教程。每个考试都包含几个主题，每个主题在 developerWorks 上都有一个自学教程。对于 LPI 考试 202 来说，包含 7 个主题，这些主题在 developerWorks 上都有对应的教程：

表 1. LPI 考试 202：教程和主题

LPI 考试 202 主题	developerWorks 教程	教程总结
主题 205	LPI 考试 202 准备（主题 205）：网络配置	介绍如何配置基本的 TCP/IP 网络，范围从硬件层（常见的以太网，modem、ISDN 或 802.11）到网络地址的路由。

主题 206	LPI 考试 202 准备（主题 206）：邮件和新闻	介绍如何将 Linux 用作邮件服务器和新闻服务器。本教程将介绍邮件传输、本地邮件过滤以及邮件列表维护软件等方面的知识，还将介绍 NNTP 协议使用的服务器软件。
主题 207	LPI 考试 202 准备（主题 207）：DNS	介绍如何使用 Linux 作为 DNS 服务器，主要是使用 BIND。本教程还将学习如何执行基本的 BIND 配置、管理 DNS 区域和保障 DNS 服务器的安全。
Topic 208	LPI 考试 202 准备（主题 208）：Web 服务	介绍如何安装和配置 Apache Web 服务器，以及如何实现 Squid 代理服务器。
主题 210	LPI 考试 202 准备（主题 210）：网络客户机管理	介绍如何配置 DHCP 服务器、NIS 客户机和服务器、LDAP 服务器和 PAM 身份验证支持。
主题 212	LPI 考试 202 准备（主题 212）：系统安全性	（本教程）介绍如何配置路由器，如何保障 FTP 服务器的安全，如何配置 SSH，以及如何执行各种其他安全管理任务。参见下面详细的 目标。
主题 214	LPI 考试 202 准备（主题 214）：网络故障诊断	敬请期待

要开始准备 1 级认证，请参考 针对 LPI 101 考试的 developerWorks 教程。要准备 2 级认证的其他考试，请参考 针对 LPI 201 考试的 developerWorks 教程。阅读 全套 developerWorks LPI 教程。

Linux Professional Institute 不为任何第三方考试准备资料或技术做担保。详情请联系 info@lpi.org。

关于本教程

欢迎阅读“系统安全性”，这是介绍 Linux 上中级网络管理的由 7 篇文章组成的教程系列中的第 6 篇。在本教程中，您将学习到与使用 Linux 作为安全敏感网络服务器有关的各种主题的内容。本文将介绍诸如路由、防火墙以及 NAT 转换之类的问题（以及对它们进行管理的工具），同时将介绍为 FTP 和 SSH 设置的安全策略。我们还将学习有关使用 tcpd、hosts.allow 以及相关工具来实现通用访问控制的内容（请回顾一下 LPI 201 考试准备（主题 209）：文件和服务共享 中的讨论）。最后，我们还将学习有关基本安全监视工具的内容，例如到什么地方查找安全资源。

与 developerWorks 201 和 202 系列中的其他教程一样，本教程旨在充当考试准备的学习指南和入门，而不是该主题的完全文档。鼓励读者参考 LPI 的 详细目标列表，并在需要时参考其他资料来补充这里的知识。

本教程是按照这个主题的 LPI 目标（objective）进行组织的。大致上说，学习目标的权值越高，在考试中出的题就越多。

表 2. 系统安全性：本教程涉及的考试目标

LPI 考试目标	目标权值	目标摘要
2.212.2 配置路由器	权值 2	配置系统实现网络地址转换（NAT、IP 封装）功能，并说明它在保护网络方面的重要性。这个目标包括配置端口重定向，管理过滤规则，以及如何防止攻击。
2.212.3 提高 FTP 服务器的安全性	权值 2	可以配置 FTP 服务器来实现匿名下载和上传。这个目标包括在允许匿名上传的情况下所应该采取的预防措施，以及配置用户的访问权限。

2.212.4 Secure shell（SSH）	权值 2	配置 SSH 守护进程。这个目标包括管理密钥、为用户配置 SSH、通过 SSH 来转发应用程序协议以及管理 SSH 登录。
2.212.5 TCP_wrappers	权值 1	配置 tcpwrappers，只允许来自某些特定主机或子网的连接发往指定的服务器。
2.212.6 安全任务	权值 3	安装并配置安全身份验证系统；对源代码进行一些基本的安全审计操作；从各个资源处接受安全警告；为了开放电子邮件中继 FTP 服务器和匿名 FTP 服务器而对服务器进行审计；安装、配置并运行入侵检测系统；应用安全补丁和 bug 修复包。

前提条件

要想从本教程获得最大的好处，那么您应该具备 Linux 的一些基础知识，并拥有一个可以用来实践本教程中介绍的命令的 Linux 系统。

其他资源

对于大多数 Linux 工具来说，查看讨论过的所有实用工具手册通常很有用。实用工具或内核之间的版本和开关（switch）可能有所不同，或者具有不同的 Linux 发行版。关于更深入的信息，Linux Documentation Project 提供了各种有用的文档，尤其是它的 HOWTO 文档。关于 Linux 网络的书籍已经出版了很多，我觉得 O'Reilly 出版的由 Craig Hunt 编写的 TCP/IP Network Administration 一书相当有用（参见 参考资料 中的链接）。

配置路由器

关于报文过滤

Linux 内核中包括了 “netfilter” 的基础设施，它允许您对网络报文进行过滤。通常这种功能会编译到基本的内核中，不过也可能会需要使用一个内核模块。不管怎样，模块的加载都应该是无缝的（例如，如果需要，运行 `iptables` 就可以加载 `iptables_filter.o`）。

报文过滤在现代 Linux 系统中是使用 `iptables` 工具来控制的；较早的系统使用的是 `ipchains`。在此之前，通常使用 `ipfwadm`。如果需要向后兼容，我们仍然可以 在最新的内核中使用 `ipchains`；不过我们通常都会倾向于使用 `iptables` 中增强的功能和改进后的语法。也就是说，`iptables` 中的大部分概念和选项都是与 `ipchains` 兼容的一些增强。

根据过滤机制的具体情况（防火墙、NAT 等），过滤和地址转换可能发生在对报文进行路由之前，也可能发生在对报文进行路由之后。在这两种情况下，可以使用相同的 `ipchains` 工具，但是要对每一种情况使用不同的规则集（“链”）—— 最基本的链是 `INPUT` 和 `OUTPUT`。但是，通过对 `FORWARD` 链进行过滤，可能对路由决定产生影响；这会导致丢包，而不是对包进行路由。

路由

告诉所有人。将其提交到：

正如使用 `iptables`（或更传统的 `ipchains`）进行的过滤那样，Linux 内核也会对自己接收到的 IP 报文进行路由。路由的过程比过滤更简单，不过二者从概念上来说是相关的。

在路由过程中，主机可以只查看目标 IP 地址，并确定自己是否知道如何将报文直接发往这个地址。如果主机既不能自己来传送这个报文，也不知道将报文转发给哪个网关，那么它会将这个报文丢弃。然而，典型的配置包含一个 “默认网关”，负责处理这些无法确定的地址。

路由信息的配置和显示可以使用 `route` 工具来实现。然而，路由可以是静态的，也可以是动态的。

使用静态路由，报文的发送是由一个路由表来确定的，路由表可以通过调用 `route` 命令及其 `add` 或 `del` 命令显式地进行配置。然而，使用 `routed` 或 `gated` 守护进程（它们会将路由信息广播给相邻的路由守护进程）来配置动态路由通常会更加有用。

`routed` 守护进程支持 Routing Information Protocol（RIP）；`gated` 守护进程增加了对很多其他协议的支持（并且可以一次使用多种协议），例如：

- Routing Information Protocol Next Generation（RIPng）
- Exterior Gateway Protocol（EGP）
- Border Gateway Protocol（BGP）和 BGP4+
- Defense Communications Network Local-Network Protocol（HELLO）
- Open Shortest Path First（OSPF）
- Intermediate System to Intermediate System（IS-IS）
- Internet Control Message Protocol（ICMP 和 ICMPv6）/Router Discovery

下面让我们来看一个相当典型的静态路由表：

清单 1. 典型的静态路由表

```
% /sbin/route
Kernel IP routing table
Destination   Gateway         Genmask         Flags Metric Ref  Use Iface
66.98.217.0   *               255.255.255.0   U     0      0      0 eth0
10.10.12.0    *               255.255.254.0   U     0      0      0 eth1
66.98.216.0   *               255.255.254.0   U     0      0      0 eth0
169.254.0.0   *               255.255.0.0     U     0      0      0 eth1
default       ev1s-66-98-216- 0.0.0.0         UG    0      0      0 eth0
```

这意味着 66.98.217/24 和 66.98.216/23 范围内的地址都会通过 `eth0` 设备直接传输。10.10.12/23 和 169.254/16 地址范围会在 `eth1` 设备上传输。其他范围内的地址都会被发送给网关 `ev1s-66-98-216-1.ev1servers.net`（这个名字是从 `route` 显示的结果中截取出来的；我们使用 `route -n` 命令就可以看到这个名字对应的 IP 地址是 66.98.216.1）。如果希望为其他地址范围添加一个不同的网关，那么可以运行下面的命令：

清单 2. 为其他地址范围添加一个新网关

```
% route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.2.1 dev eth0
```

对于一台网关机器来说，我们通常会希望运行动态路由，此时要使用 `routed` 或 `gated` 守护进程，它可以提供少量的静态路由。`routed` 守护进程是通过 /etc/gateways 的内容进行配置的。`gated` 守护进程是比较新的方式，它具有更多的功能（正如前面介绍的一样），它是通过 /etc/gated.conf 进行配置的。通常，不管我们使用其中的哪一个，都会希望在启动脚本中加载它。我们不能在一台机器上同时运行 `routed` 和 `gated`，因为这样会导致无法预期甚至是不希望的结果。

使用 iptables 过滤报文

Linux 内核为 IP 报文保存了一个过滤规则表，它们会构成一种状态机。这些依次处理的规则集称为“（防火墙）链”。当一个链符合某个条件时，就会有一个可能的操作将控制权交出，从而可以向状态机一样处理另外一个链。在添加任何规则或状态之前，系统中已经自动出现了 3 个链：`INPUT`、`OUTPUT` 和 `FORWARD`。`INPUT` 链是发往主机机器的报文所要经过的地方，它们也可能来自本地应用程序进程。`FORWARD` 链是发往其他机器的报文要经过的地方，此时假设我们已经启用了转发功能，并且路由系统知道如何转发这个报文。在本地机器上生成的报文被发送到 `OUTPUT` 链中进行过滤 —— 如果它通过了 `OUTPUT` 链（或任何相关链）的过滤器，就会通过网络接口路由出去。

规则（rule）可以采取的一种操作是 `DROP####` 报文；在这种情况下，不会为这个报文执行其他规则处理或状态转换操作。但是，如果报文没有被丢弃，那么就会对链中的下一条规则进行检查，查看它与这个报文是否匹配。在某些情况下，满足某条规则会将处理过程转入另一个不同的链以及它的规则集中。规则和规则所在的链的创建、删除或修改都是使用 `iptables` 工具来执行的。在早期的 Linux 系统中，相同的功能是使用 `ipchains` 实现的。这两个工具以及更早的 `ipfwadm` 背后的概念都是类似的，但是我们在这里讨论的是 `iptables` 的语法。

规则指定了一组报文可能符合的条件，以及如果报文符合这些条件后应该执行哪些操作。正如前面介绍的那样，一个通用的操作是 `DROP` 报文。例如，假设我们希望（由于某种原因而）禁用本地回环地址上的 `ping` 操作（ICMP 接口）。那么我们可以使用下面的命令来实现这项功能：

清单 3. 禁用本地回环接口上的 ICMP

```
% iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP
```

当然，这只是一条不太好的规则，我们可能在经过测试之后又想删除它，例如：

清单 4. 删除这条规则

```
% iptables -D INPUT -s 127.0.0.1 -p icmp -j DROP
```

使用 `-D` 选项删除规则时可以使用与添加规则完全相同的参数，规范可以通过规则编号指定（此时我们必须先确定这条规则的编号），如下所示：

清单 5. 指定规则编号，这样也可以正常删除规则

```
% iptables -D INPUT 1
```

更有意义的规则可能会查看报文中的源地址和目标地址。例如，假设一个有问题的远程网络正试图使用我们网络上某个特定子网中的服务。我们可以在网关/防火墙机器上使用下面的命令阻塞它：

清单 6. 阻塞网关/防火墙机器

```
% iptables -A INPUT -s 66.98.216/24 -d 64.41.64/24 -j DROP
```

这样做会阻断 `66.98.216.*` IP 范围的机器与本地 `64.41.64.*` 子网上的任何机器进行通信。当然，选择某个特定 IP 范围加入黑名单对于保护手段来说是非常有限的。更可能的情况是只允许 来自某个特定 IP 范围的机器访问本地子网：

清单 7. 允许指定的 IP 范围访问本地子网

```
% iptables -A INPUT -s ! 66.98.216/24 -d 64.41.64/24 -j DROP
```

在这种情况下，只有 `66.98.216.*` IP 范围内的机器可以访问这个指定的子网。另外，我们可以为某个地址使用符号名，并指定哪个特定的协议要进行过滤。我们也可以选择一个特定的网络接口（例如 `eth0`）进行过滤，但是这通常不会有太大用处。例如，如果只想让某个远程网络访问本地 Web 服务器，那么可以使用：

清单 8. 让某个远程网络访问本地 Web 服务器

```
% iptables -A INPUT -s ! example.com -d 64.41.64.124 -p TCP -sport 80 -j DROP
```

我们可以使用 `iptables` 指定其他一些选项，包括根据 TCP 标记来设置允许通过或过滤掉的报文数目比率。有关的更多详细信息，请参见 `iptables` 的手册。

用户定义的链

我们已经介绍了向自动链中添加规则的基础知识。但是 `iptables` 中更多的可配置内容是通过添加用户定义的链并在符合某种模式时对它们进行细分来实现的。新链是使用 `-N` 选项定义的；我们已经查看了使用特殊目标 `DROP` 进行细分的情况。`ACCEPT` 也是一个特殊的目标，其意义显而易见。另外，特殊目标 `RETURN` 和 `QUEUE` 都可以使用。第一个目标表示停止处理某个给定的链，并返回自己的父链/调用者。`QUEUE` 处理器允许我们将报文传递给用户空间的进程，以进行进一步的处理（这可能是记录日志、修改报文内容或进行比 `iptables` 所支持范围更为广泛的处理）。Rusty Russell 的 “Linux 2.4 Packet Filtering HOWTO” 中的这个简单示例就是添加用户定义链的一个很好的例子：

清单 9. 添加用户定义的链

```
# Create chain to block new connections, except established locally
% iptables -N block
% iptables -A block -m state --state ESTABLISHED,RELATED -j ACCEPT
% iptables -A block -m state --state NEW -i ! ppp0 -j ACCEPT
% iptables -A block -j DROP  # DROP everything else not ACCEPT'd
# Jump to that chain from INPUT and FORWARD chains
% iptables -A INPUT -j block
% iptables -A FORWARD -j block
```

注意 `block` 链会接收（`ACCEPT`）有限种类的报文，最后一条规则会丢弃（`DROP`）前面不被接收（`ACCEPT`）的所有内容。

一旦建立起某些链之后，不管我们是向自动链中添加规则，还是添加用户定义的链，都可以使用 `-L` 选项来查看当前规则。

网络地址转换与防火墙的对比

前面介绍的例子大部分都属于防火墙规则的范畴。但是网络地址转换（NAT）也可以使用 `iptables` 进行配置。

从根本上来说，NAT 是一种使用连接将来自本地子网地址的封装报文在发送（在 `OUTPUT` 链上）之前作为外部 WAN 地址进行跟踪的方法。执行 NAT 的网关/路由器需要记住哪台本地机器连接到了哪台远程机器上，当报文从远程机器上返回时，它需要对这种地址转换反向进行解析。

尽管从过滤器的角度来看，我们可以简单地认为 NAT 并不存在。我们指定的规则应该使用 “真正的” 本地地址，而不必关心 NAT 如何封装它之后将其呈现给外部世界。要启用封装，例如基本 NAT 中实现的那样，只需要使用下面的 `iptables` 命令即可。要使用这个命令，则需要确保已经加载了内核模块 iptables_nat，并且启用了 IP 转发功能：

清单 10. 启用封装

```
% modprobe iptables_nat    # Load the kernel module
% iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
% echo 1 > /proc/sys/net/piv4/ip_forward   # Turn on IP forwarding
```

这种功能称为源 NAT —— 我们修改的是外发报文的地址。目的 NAT（DNAT）也可以用来启用端口转发、负载共享和透明代理。在这些情况下，传入的报文都要进行修改，然后才能进入相关的本地主机或子网。

但在许多情况下，当用户或管理员谈到 NAT 时，他们所说的都是源 NAT。如果要配置目的 NAT，则需要指定 `PREROUTING`，而不是 `POSTROUTING`。对于 DNAT 来说，报文是在路由之前进行地址转换的。

提高 FTP 服务器的安全性

FTP 服务器

很多不同的 FTP 服务器都可以在 Linux 上使用，不同的发行版可能会提供不同的服务器。通常，不同服务器的配置并不相同，不过大部分都可以遵循类似的配置准则。

一种非常流行的 FTP 服务器是 vsftpd（Very Secure FTP 守护进程）。ProFTP 的使用也非常广泛，另外还有 wu-ftpd 和 ncftpd。

对于很多目的来说，FTP 并不是真正需要的。例如，对于那些在某台服务器机器上拥有帐号的用户来说，要想安全地传输文件，只需使用 `scp`（安全复制）即可，scp 依赖于底层的 SSH 安装，不过它在很大程度上模仿了我们熟悉的 `cp` 命令。

vsftpd 的配置文件是 /etc/vsftpd.conf。其他的 FTP 服务器也使用了类似的文件。

FTP 配置选项

下面是 /etc/vsftpd.conf（如果我们使用的是其他 FTP 服务器，系统中也可能存在这个文件）中我们需要记住的几个选项：

- `anonymous_enabled`：允许匿名用户使用用户名 “anonymous” 或 “ftp” 进行登录。
- `anon_mkdir_write_enable`：允许匿名用户创建目录（在大家都可写的父目录中）。
- `anon_upload_enable`：允许匿名用户上传文件。
- `anon_world_readable_only`：默认值为 “YES”；修改这个值不是什么好主意。只允许匿名 FTP 访问全局可读文件。
- `chroot_list_enable`：指定一组用户（在 /etc/vsftpd.chroot-list 中给出）在登录时不能从自己的主目录 “chroot” 到的地方。
- `ssl_enable`：支持 SSL 加密的连接。

有关的更多具体选项，请参阅 FTP 服务器的手册。通常，运行 FTP 服务器非常简单，只需修改配置文件并在初始化脚本中运行服务器即可。

安全 shell（SSH）

客户机和服务器

几乎每台 Linux 机器（以及大部分其他操作系统）都应该有一台安全 shell（SSH）客户机。通常我们使用的都是 OpenSSH 版，但是有时会使用很多兼容的 SSH 客户机。只要一台 SSH 客户机就足以连接到主机上，因此正确配置 SSH 服务器的巨大安全问题就浮现了。

由于客户机会发起到服务器的连接，因此这台客户机会主动选择信任服务器。正如拥有一台 SSH 客户机并不意味着允许它进入某台机器进行访问一样；这也不会带来什么缺陷。

配置服务器也不是特别复杂；服务器守护进程被设计用来启用并强制采用良好的安全实践。但是有一点非常清楚：它是一台服务器，将与基于客户机与服务器之间的请求的客户机共享资源。

SSH 协议有两个版本：版本 1 和版本 2。在现代系统中，通常倾向于使用该协议的版本 2，但是通常会在客户机和服务器之间维护与版本 1 之间的向后兼容性（除非这种功能已经使用配置选项禁用了）。这使我们可以连接到日益少见的只使用版本 1 的系统上。

版本 1 和版本 2 协议使用了一些不同的配置文件。对于协议版本 1 来说，客户机首先会使用 ssh-keygen 创建一个 RSA 密钥对，并将自己的私钥保存到 $HOME/.ssh/identity.pub 中。这个相同的 identity.pub 应该添加到远程的 $HOME/.ssh/authorized_keys 文件的后面。

显然，这是一个鸡和蛋的问题：在具有访问权限之前，如何将一个文件复制到远程系统上呢？幸运的是，SSH 还支持 fallback 身份验证方法，这样就可以在加密线路上发送密码，并对密码进行常见的远程系统登录测试（例如，用户帐号必须存在，必须提供正确的密码）。

协议 2 支持 RSA 和 DSA 密钥，但是 RSA 身份验证与协议 1 并不完全相同，而是有了一定的增强。对于协议 2 来说，私钥保存在 $HOME/.ssh/id_rsa 和 $HOME/.ssh/id_dsa 中。协议 2 还可以支持其他很多机密和完整性算法：AES、3DES、Blowfish、CAST128、HMAC-MD5、HMAC-SHA1 等。服务器可以配置成优先选择算法和 fallback 身份验证的次序。

对于密钥信息之外的其他通用配置选项来说，客户机会将自己的密钥保存到 /etc/ssh/ssh_config 中（或保存在 /$HOME/.ssh/config 中）。客户机选项也可以使用 `-o` 选项进行配置；比较特别的通用选项是 `-X` 或 `-x`，可以用它们来启用或禁用 X11 转发。如果启用 X11 转发，X11 端口就会通过 SSH 设置隧道，从而启用加密的 X11 连接。

诸如 `scp` 之类的工具通过 SSH 使用类似的端口转发功能。例如，在我使用的本地机器上，可以在本地显示设备上启动一个远程的 X11 程序（在本例中，远程机器与我的机器在相同的本地子网中）：

清单 11. 启动远程 X11 程序

```
$ which gedit  # not on local system
$ ssh -X dqm@192.168.2.2
Password:
Linux averatec 2.6.10-5-386 #1 Mon Oct 10 11:15:41 UTC 2005 i686 GNU/Linux
No mail.
Last login: Thu Feb 23 03:51:15 2006 from 192.168.2.101
dqm@averatec:~$ gedit &
```

配置服务器

`sshd` 守护进程，尤其是 OpenSSH 版的，可以通过不安全的网络在两个不可信的主机之间启用安全加密通信。基本的 `sshd` 服务器通常是在初始化过程中启动的，并为每个客户机连接派生一个新守护进程来监听客户机的连接。所派生的守护进程负责处理密钥交换、加密、身份验证、命令执行和数据交换的功能。

与使用客户机工具一样，`sshd` 服务器可以在命令行中接收各个选项，但是通常这些选项都是通过 /etc/ssh/sshd_config 文件进行配置的。它还会使用其他很多配置文件。例如，访问控制 /etc/hosts.allow 和 /etc/hosts.deny 文件。对于客户机端来说，密钥的存储方式类似，也是保存在 /etc/ssh/ssh_host_key（协议 1）、/etc/ssh/ssh_host_dsa_key、/etc/ssh/ssh_host_rsa_key 中，公钥保存在 /etc/ssh/ssh_host_dsa_key.pub 和相关文件中。另外，与客户机一样，我们可以首先使用 ssh-keygen 生成密钥。有关这些配置文件以及如何将所生成的密钥复制到适当的文件中的详细信息，请参见 `sshd` 和 `ssh-keygen` 的手册。

很多配置选项都位于 /etc/ssh/sshd_config 中，默认值通常都很适合（也是相当安全的）。下面是几个值得注意的选项：

- `AllowTcpForwarding` 启用或禁用端口转发（隧道），默认值为 “YES”。
- `Ciphers` 控制了要使用的加密算法的清单和次序。
- `AllowUsers` 和 `AllowGroups` 可以接收通配符模式，并让我们可以控制哪个用户可以尝试进一步进行身份验证。
- `DenyGroups` 和 `DenyUsers` 与上面的两个选项正好相反，这与我们预期的一样。
- `PermitRootLogin` 使根用户可以使用 SSH 登录到机器中。
- `Protocol` 使我们可以指定是否接受两个版本的协议（如果不能同时接受这两个版本的协议，则应该接受哪一个版本）。
- `TCPKeepAlive` 可以很好地查看是否丢失了 SSH 连接。如果启用了这个选项，就会发送 “keepalive” 消息来连接进行检查，但是如果在路由中出现暂时性的错误，则会导致连接断开。

SSH 隧道

OpenSSH 允许我们创建一个隧道，将另外一个协议封装到加密后的 SSH 通道中。这项功能在默认情况下是在 `sshd` 服务器上启用的，但也可以使用命令行和配置文件选项来禁用它。假设我们启用了这项功能，那么客户机都可以模拟自己希望将哪个端口/协议用于连接。例如，要为 telnet 创建一个隧道，可以使用以下命令：

清单 12. 为 telnet 建立一个隧道

```
% ssh -2 -N -f -L 5023:localhost:23 user@foo.example.com
% telnet localhost 5023
```

当然，这个例子没有太大的价值，因为 SSH 命令 shell 也可以像 telnet shell 一样实现相同的功能。但是我们可以以类似的方式创建一个 POP3、HTTP、SMTP、FTP、X11 或其他协议连接。基本概念是可以将一个特定的本地主机端口用作模拟的远程服务端口，真正的通信报文会通过这个 SSH 连接以加密的形式进行传输。

我们在这个例子中使用的选项有：

- `-2`（使用协议 2），
- `-N`（只没有命令/通道）
- `-f`（在后台使用 SSH），以及
- `-L,`（说明通道的格式是 “localport:remotehost:remoteport”）。

当然还要指定服务器（和用户名）。

TCP_wrappers

什么是 “TCP_wrappers”？

关于 TCP_wrappers，我们要知道的第一件事情是不应该使用它，现在它的维护也已不再活跃。不过，我们仍然可能从目前仍在运行的一些老系统上的 TCP_wrappers 中找到 tcpd。在它存在的时间内，这是一个非常好的应用程序，但是其功能已经被 iptables 和其他工具所取代。TCP_wrappers 的一般用途是监视和过滤针对 SYSTAT、FINGER、FTP、TELNET、RLOGIN、RSH、EXEC、TFTP、TALK 和其他网络服务的传入请求。

TCP_wrappers 可以使用多种方式进行配置。一种方式是为其他服务器替换 tcpd，并在 tcpd 完成日志记录和过滤之后提供一些参数，将控制权传递给特定的服务器。另外一种方式是留下网络守护进程，并修改 inetd 配置文件。例如，下面这个条目：

```
tftp dgram udp wait root /usr/etc/tcpd in.tftpd -s /tftpboot
```

导致传入的 tftp 请求用进程名 `in.tftpd` 通过封装程序（tcpd）来运行。

安全任务

这个目标包含很多任务，这些任务对于维护一个安全网络来说非常重要，在本教程非常有限的篇幅中，无法对这些任务进行详细介绍。因此我们建议您花一些时间来熟悉这一节介绍的资源和工具。

为了了解安全问题和补丁而值得对其进行监视的 Web 站点包括：

- 安全焦点新闻：Security Focus Web 站点是报告和讨论安全问题和特定缺陷的一个最好的站点。这个站点包括了几个可以订阅的新闻和警告，以及一些通用的栏目和可搜索的 bug 报告。
- Bugtraq 邮件列表：这是一个完整的中等描述邮件列表，详细讨论并公布了一些计算机安全缺陷：它们是什么样的安全缺陷、如果防止以及如何修复。
- CERT Coordination Center：位于 Carnegie Mellon University，CERT 有很多与 Security Focus 站点类似的顾问，它们重点关注的是教程和应用指南。跟踪多个这种站点是确保在对 OS、发行版以及特定工具或服务器产生影响的安全事件保持最新了解的一个好方法。
- Computer Incident Advisory Capability：CIAC Information Bulletins 分布在 Department of Energy 社区中，用于通知计算机安全缺陷和推荐操作的站点。类似地，CIAC Advisory Notices 负责提醒一些有严重缺陷的站点，一旦可能就给出要应用的解决方案。CIAC Technical Bulletins 涉及了技术安全问题和对时间不太敏感的一些特性的分析。
- 安全开放邮件中继的信息：在有邮件服务器的系统上，常见的一种缺陷是不能正确地保障系统安全，防止通过垃圾邮件和欺骗性邮件进行欺骗。Open Relay Database 提供了有关特定于安全性的邮件工具、开放中继测试在线工具以及知名问题服务器数据库（如果站点管理员希望，可以用它来设置黑名单）的教程。

告诉每一个人。提交到：

我们可以考虑的用来监视安全性的工具包括：

- Open Source Tripwire：这是一个安全和数据集成工具，可以用来对特定的文件变化进行监视并产生警报。
- `scanlogd`：一个 TCP 端口扫描检测工具。
- Snort：使用规则驱动的语言来实现网络入侵预防和检测功能；它使用了基于签名、协议和反常事件的检测方法。

参考资料

学习

- 请复习 developerWorks 上的 LPI 考试准备教程系列，学习 Linux 的基础知识，并准备参加系统管理员认证考试。
- 在 LPIC Program 中，可以查看针对 Linux Professional Institute Linux 系统管理认证的三个认证级别的任务列表、示例问题和详细目标。
- 由 Craig Hunt 撰写的 TCP/IP Network Administration, Third Edition（O'Reilly，2002 年 4 月）是有关 Linux 网络的一个好资源。
- 有关更详细的信息，请访问 Linux Documentation Project，这里有很多有用的文档，尤其是 HOWTO 文档。
- 在 developerWorks Linux 专区 中可以找到为 Linux 开发人员准备的更多资源。
- 随时关注 developerWorks 技术事件和网络广播。

获得产品和技术

- 使用 IBM 试用版软件 构建您的下一个开发项目，可以从 developerWorks 上直接下载这些软件。

讨论

- 世界上超过 700 个 Linux 用户组 列表可以帮助您找到本地或远程的 LPI 考试学习小组。
- 通过参与 developerWorks blogs 加入 developerWorks 社区。

关于作者

David Mertz

从 2000 年开始，David Mertz 就一直在为 developerWorks 专栏 Charming Python 和 XML Matters 撰稿。您可以阅读他撰写的书籍 Text Processing in Python。有关 David 的更多信息，请访问其 个人主页。