语　　法：Auth_MSQLgrp_field　mSQL 含有组名的字段的名称

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　　盖：AuthConfig

本命令为 Auth_MSQLgrp_table 命令指定的表设置组名字段的名称。

10.2.4.8　Auth_MSQL_nopasswd 命令

语　　法：Auth_MSQL_nopasswd on | off

缺 省 值：off

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　　盖：Authconfig

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

如果数据库中口令字段为空，当将该命令的值设为“on”时，mod_auth_msql 模块将不比较口令。缺省值能保证 mSQL 表中口令字段为空的任何用户不可以访问受限目录。

10.2.4.9　Auth_MSQL_Authoritative 命令

语　　法：Auth_MSQL_Authoritative on | off

缺 省 值：on

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　　盖：Authconfig

缺省设置使 mod_auth_msql 模块成为必须通过的认证模块。换言之，当用户提供的用户名/口令不能通过这一模块认证时，低一层的认证模块，如标准 mod_auth 模块，就不继续进行认证。

10.2.4.10　Auth_MSQL_EncryptedPasswords 命令

语　　法：Auth_MSQL_EncryptedPasswords on | off

缺 省 值：on

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　　盖：Authconfig

该命令的缺省值告诉 mod_anth_msql 模块需要用标准的 crypt( )函数对口令进行加密。如将缺省值设为“off”，口令将以纯文本的形式表示。

10.2.5　使用 mod_auth_anon 模块

本模块允许以匿名方式访问需要认证的区域。如果熟悉匿名 ftp 服务器，就会发现本模块的设置和匿名 ftp 很相似。所有用户可以使用“anonymous”作为用户 ID，并以电子邮件地址作为口令进行访问。所送入的电子邮件地址存储在日志文件中，可用它来跟踪用户，甚至可以用来建立预期的客户邮递表。

和前面讨论的 DBM 模块类似，必须像下面这样将该模块加到配置文件中：

```
AddModule modules/standard/mod_auth_anon.o
```

接着运行配置脚本和 make 实用程序来建立 httpd 可执行文件。

做完以上工作，即可使用 mod_auth_anon 模块。让我们仔细研究该模块的一些命令，并举一个使用这一模块的例子。

10.2.5.1 Anonymous 命令

语　　法：Anonymous user user …

缺 省 值：没有

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　　盖：Authconfig

使用这一命令可指定一个或多个可以访问受限区域的用户名。最好将“anonymous”放在选择的用户列表中，因为在匿名访问中该用户名用得最广泛。如果所选用的用户名中有空格字符，一定要将这个用户名加引号，例如：

```
Anonymous "Unregistered User" anonymous
```

或

```
Anonymous 'Unregistered User' anonymous
```

用户名字符串不区分大小写。

10.2.5.2 Anonymous_Anthoritative 命令

语　　法：Anonymous_Anthoritative on | off

缺 省 值：off

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　　盖：Authconfig

当设为“on”时，匿名认证成为目录唯一的认证模式。换言之，当认证失败时，其他模块不会接着进行认证。

10.2.5.3 Anonymous_LogEmail 命令

语　　法：Anonymous_LogEmail on | off

缺 省 值：on

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　　盖：Authconfig

当将本命令设置为“on”的，在浏览器的弹出认证窗口的口令字段中键入的所有信息都将记录到 Apache 访问日志文件中。

10.2.5.4 Anonymous_MustGiveEmail 命令

语　　法：Anonymous_MustGiveEmail on | off

缺 省 值：on

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　　盖：AuthConfig

当将这一命令设为“on”时，它使模块拒绝没有以 e-mail 地址形式提供口令的访问请求。然而并不能信任 E-mail 地址，因为根本无法检查发来的 e-mail 地址到底是谁的。

10.2.5.5 Anonymous_NoUserID 命令

语　　法：Anonymous_NoUserID on | off

缺 省 值：off

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　　盖：AuthConfig

如果允许浏览器的弹出窗口中的用户名字段为空，可以将此命令设置为“on”，否则用户名必须与 Anonymous 命令指定的用户名匹配才允许登录。

10.2.5.6　Anonymous_VerifyEmail 命令

语　　法：Anonymous_VerifyEmail　on | off

缺 省 值：off

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　　盖：AuthConfig

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

当将本命令的值设为“on”时，它要求口令必须是有效的 e-mail 地址。然而这种检查的有效性是有限的。本模块仅检该口令字段中是否有“@”和“.”字符，如果口令中有这些字符，它就接受。

10.2.5.7　匿名访问限制示例

下面的配置表示如何用前述命令对一个目录提供匿名访问：

```
Anonymous_NoUserId off
Anonymous_MustGiveEmail on
Anonymous_VerifyEmail on
Anonymoun_LogEmail on
Anonymous anonymous guest "I do not know"
AuthName Use 'anonymous' & Emial address for guest entry
AuthType Basic
require valid-user
```

10.2.6　使用 mode_auth_external 模块

到目前为止，已讨论了可以用标准模块实现的各种认证方法。但是，如果需要使用第三方提供的认证模式该怎样办呢？ 这种情况下，可使用 mod_auth_external 模块来使用外部的认证。如果将 mod_auth_external 模块编译到服务器可执行文件中，Apache 就可以使用外部认证程序。

为了安装 mod_auth_external 模块，首先应将该模块加到配置文件中。用文本编辑程序将下列行加到 Apache 配置文件中：

```
AddModule modules/standard/mod_auth_external.o
```

如果打算将该模块放到 Apache 源目录树以外的任何目录中，则要相应地改变前面的路径。

现在运行配置脚本建立新的 Makefile，然后运行 make 实用程序建立新的 httpd 可执行文件。建立可执行文件后，用 httpd -l 命令列出它所包含的模块：

```
httpd  -l
```

如果在该命令产生的列表中有 mod_auth_external.c 模块，则表明已成功安装了该模块。如果喜欢为该模块写自定义的函数，可按以下步骤去做（如果对写函数不感兴趣的话，可跳过以下几个步骤）。

步骤1：在模块中支持使用自定义函数

编辑 mod_auth_external.c 文件，去掉以下几行的注释：

```
# define _HARDCODE_
# ifdef HARDCODE
# inclue "your_function_here.h"
# endif
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

使用用户定义的认证函数的头文件代替上面代码中的“your_function_here.h”，例如将认证函数源文件称为“my_auth.c”，并且该函数原型存在 “my_auth.h”文件中，那么应将下面几行代码加到“mod_auth_external.c”文件中：

```
# define _HARDCODE_
# ifdef HARDCODE
# include "my_auth.h"
# endif
```

确保源码和“mod_auth_external.c”文件存在于同一目录中。

步骤2：在模块中对自定义函数编码

在“mod_auth_external.c”文件的中间找到大的注释段，然后按那里的指示做。函数应该像下面那样的形式开始：

```
int
function_name (char *user_name, char *user_passwd, char
*config_path){
the function call in mod_auth_exterual.c shonld look
something like:

if (strcmp(check_type, "<type>")==0){
code = function_name(c->user, sent_pw, config_file);
}
```

Note

不要使用 exit( )或其他能引起函数不正常退出或退出内核的调用。这类调用会完全退出 httpd，并向浏览器发出“no data”消息。用 return( ) 函数来代替 exit( )。

步骤3：保存文件并编译 Apache

在 Apache 源代码目录中运行 make 实用程序，如果编译不能通过，那么需调试代码。任何时候当修改了源代码文件（如 my_auth.c），并没有对 mod_auth_external.c 作任何改变，这时应该使用 touch 实用程序改变 mod_auth_external.c 文件的修改时间。

如果一切顺利，就可编译 Apache，并用新版本替换旧的 Apache 服务器。

安装了 mod_auth_external 模块后，即可访问三种外部认证机制。

第一种认证机制是 System( )调用，它是缺省方法。使用这种方法，模块调用用户（即 Apache 管理员）定义的认证模式，并传给它 USER 和 PASS 两个环境变量。从客户端键入的用户名和口令分别用这两个环境变量来传递。外部认证程序可以读环境变量，并执行必要的检查以决定是否允许该用户访问。该模块要求外部认证程序在成功时返回退出码 0，否则返回 1。

在大多数 Unix 系统上，可以用 ps 实用程序来检查正在运行的进程使用哪些环境变量。例如：

```
ps -auxwe
```

在 Linux 系统上，上面的命令将显示所有运行程序的环境变量。当认证检查程序在运行时，系统上的某些人就可能性发现 USER 和 PASS 变量的值是什么。这是系统的安全漏洞。

超星阅览器提醒 使用超星阅览器请尊重相关版权!

第二种外部认证方法是 Pipe( )调用。该调用和 System( )调用相似，但在这种方法中，模块将从客户端得到的用户名和口令写到外部认证程序的标准输入（STDIN）中。数据以 key=value 对的方式存储，其中 USER=用户名，PASS=口令。

第三种外部认证方法是自定义函数调用。在这种方法中，模块调用用户定义的 C 函数，并传给它用户名、口令和配置文件名。这种方法允许开发人员给已有的认证应用程序写定制的网关接口，或者开发新的认证应用程序。下面是这类函数例子中的两个：

- mod_auth_external_radius.c——使用公开的 Merit Radius 源码的 Radius 客户程序。
- mod_auth_external_sybase.c——可以查询 Sybase 数据库的函数，它对指定的用户名验证口令。

现已经知道有那些可用的外部认证方法了，接着讨论该模块提供的命令并举一个使用该模块的例子。

10.2.6.1 AddExternalAuth 命令

语法（系统调用和管道）：AddExternalAuth 关键字 认证程序路径

语法（C 函数）：AddExternalAuth 关键字 类型：配置文件路径

使用范围：服务器配置

该命令将关键字和指定的用户认证程序联系在一起，例如：

```
AddExternalAuth archive_auth /usr/local/bin/authcheck
```

它将 archive_auth 关键字和外部认证程序/usr/local/bin/authcheck 联系在一起。这一命令用于自定义函数的例子如下：

```
AddExternalAuth archive_auth RADIUS
```

其中 RADIUS 是自定义函数名。如果该函数需要配置文件，可以用下述方式传送给该函数：

```
AddExternalAuth archive_auth RADIUS: /usr/local.raddb
```

由客户端输入的用户名、口令和可选的配置文件名（仅适用于自定义函数）传递给外部用户认证程序。在缺省条件下，外部认证程序以 USER 和 PASS 环境变量的形式得到用户名和口令。如果用 SetExternalAuthMothod 命令设置了管道模式，用户名和口令将以 USER=用户名和 PASS=口令的形式作为 key=value 对传递。对于自定义函数，数据以 C 函数参数的形式传递。

如果外部用户认证程序返回 0，就认为成功地通过了认证。非零返回值则说明认证失败或程序执行错误。

通过在 srm.conf 文件中提供多个命令，可以在同一个服务器中使用多种认证方式。这只需保证每个命令都有不同的关键字。

10.2.6.2 AddExternalGroupAuth 命令

语法（系统调用和管道）：Add ExternaGroupAuth 关键字 到认证程序的路径

语法（函数）：AddExternalGroupAuth 关键字 类型：到配置文件的路径

使用范围：服务器配置

该命令将关键字与指定的组认证程序联系在一起。下面的例子将 archive_auth 关键字和名为/usr/local/bin/groupcheck 的外部认证程序联系在一起：

超星阅读器提醒你：使用本复制品请尊重相关知识产权！

```
AddExternalGroupAuth archive_auth /usr/local/bin/groupcheck
```

将用户名、组名和可选的配置文件名（仅适用于自定义函数的认证）以不同方式传递给组外部认证程序。在缺省情况下，外部认证程序以 USER 和 PASS 环境变量的形式收到用户名和口令。如果用 SetExternalGroupMethod 命令指定了管道模式，用户名和口令以 USER=用户名和 GROUP=组名的形式作为 key=value 对传递。对于自定义函数，数据以 C 函数的参数形式传递。

如果外部组身份认证程序返回 0，则认为成功地通过了认证。非零返回值则说明认证失败或认证程序执行出错。

10.2.6.3 SetExternalAuthMethod 命令

语　　法：SetExternalAuthMethod 关键字 方法

使用范围：服务器配置

本命令设置向外部认证程序传递数据的方法，目前可以使用三种方法。

第一种方法是环境变量。这种方法将用户名和口令以环境变量的方式传递，USER 环境变量包含用户名，PASS 环境变量包含口令。本方法是缺省方法。

考虑下面的例子：

```
AddExternalAuth archive_auth /usr/local/bin/authcheck
SetExternalAuthMethod archive_auth enviroment
```

在本例中，/usr/local/bin/authcheck 程序将通过环境变量 USER 和 PASS 接收客户端键入的用户名和口令。

第二种方法是函数。当使用自定义函数来处理认证时使用这种方法。

考虑下面的例子：

```
AddExternalAuth avchive_auth RADIUS:
SetExternalAuthMethod archive_auth funtion
```

其中使用了函数 RADIUS，该函数以参数形式接收数据。

第三种方法是管道。这种方法通过管道向外部认证程序的标准输入设备（STDIN）发数据。数据以 key=value 的形式发送，其中 USER=用户名和 PASS=口令。

考虑下面的例子：

```
AddExternalAuth archive_auth /usr/local/bin/authcheck
SetExternalAuthMethod archive_auth pipe
```

例中/usr/local/bin/authckeck 程序将通过管道从标准输入设备（STDIN）接收客户键入

的用户名和口令，其中 USER=用户名和 PASS=口令。

10.2.6.4 SetExternalGroupMethod 命令

语法：SetExternalGroupMethod 关键字 方法

本命令设置用于向外部组身份认证程序传递数据的方法。目前可以使用三种方法：环境变量，函数和管道。详细情况和示例可参照 SetExternalAuthMethod 命令。

10.2.6.5 AuthExternal 命令

语法（系统调用或管道）：AddExternalAuth 关键字

语法（函数）：AddExternalAuth 函数名：配置文件路径。

使用范围：目录，针对目录的访问控制文件（.htaccess）

本命令用在目录中或针对目录的访问控制文件（.htaccess）中，用来通知 Apache 用于该目录的认证程序是什么。服务器匹配关键字或函数名（在自定义认证函数的情况下）以决定采取什么行动。

例如：

```
AuthExternal archive_auth
```

如果在 .htaccess 文件中出现这行命令，Apache 将运行和这一关键字对应的用户认证程序。

10.2.6.6 GroupExternal 命令

语法（系统调用或管道）：GroupExternal 关键字

语法（函数）：GroupExternal 函数名：配置文件路径

范围：目录，针对目录的访问控制文件（.htaccess）

本命令用在目录中或针对目录的访问控制文件中。它通知 Apache 用于组认证的是哪个外部组认证程序。服务器匹配关键字或函数名（对自定义函数）以决定采取什么行动。

例如在 .htaccess 文件中发现下列行，Apache 将运行与该关键字对应的组认证程序。

```
GroupExternal archive_auth
```

10.2.6.7 示例：使用 Perl 脚本作为外部用户认证程序

在本例中，将看到如何使用简单的 Perl 脚本提供用户认证服务，为了使本例更有意义（也许更有说服力），使用的 Perl 脚本利用在大多数 Unix 系统中均能找到的/etc/passwd 和/etc/group 文件来实现认证。

很多人会说：在基于 Web 的认证中使用/etc/passwd 和/etc/group 并不是一个高明的想法。我也倾向于同意这种观点。然而，对于受到良好保护的企业网环境（如果没有连上 Internet 就更好）来说，这似乎是顶刮刮的了。任何在 Web 服务器机器上有 Unix 帐号的人均可使用列表 10-2 的 Perl 脚本来进行认证。

列表 10-2： authcheck 脚本

```
#!/usr/local/bin/perl

#
# Script Name: authcheck
```

```
# Purpose:a simple /etc/passwd and /etc/group based
# authenticator script
#
# Chapter 10 Listing 10-2
##########################################################
# Variables and assignments
my $LOG_FILE = '/tmp/authcheck.log';  # log file to be used
my $this_user = $ENV{USER};  # get username from %ENV
my $this_pwd = $ENV{PASS};   # get password from %ENV

# If the user does not enter a username or password exit
# with a non-zero value to let the mod_auth_external
# know that the authentication has failed.
#
exit 1 lf($this_user eq " " | | $this_pwd eq " " );

# We only want to allow users of a certain group to be able
# to authenticate via Web When creating this group in
# your /etc/group, remember to keep power users out
# of it. For example NEVER put the 'root' user in
# this group!
#
my $WEB_USER_GROUF = 'web_users';

# Get the current time and store it in variables
my ($sec, $min, $hr, $mday, $mon, $year, $wday, $yday, $isdst)
=localtime(time);
# Increment month to take care of the 0-11 range used by localtime
$mon++;
# If the user is a member of the specified Web user group,
# and her password is valid then we will consider the
# authentication to be a success
#
if(isMember($this_user,$WEB_USER_GROUP) &&
validPassword($this_user,$this_Pwd)){

# Log the successful authentication even. However note
# that since each request from a user will go through
# this authentication code, logging successful authentication
# might create a performance issue. If you do not want
# to log successful logins, comment the following line
#
  & log("$mon/$mday/$year $hr:$min:$sec  -$this_user login
successful.");

# Exit with the success code value so that mod_auth_external
```

```
# will consider the authentication to be a complete success.
exit 0;
    }
else{
# well, the user didn't pass the checks so log
# the attempt.
     & log("$mon/$mday/$year $hr:$min:$sec - $this_user login
failed: invalid password ($this_pwd).");

# Exit with a non-zero code to tell mod_auth_external
# that the user has failed authentication
    exit 1;
    }

sub validPassword{
#
# Purpose: this sub routine checks the user's password
# against the password stored in the /etc/passwd file.

# Get the username and the user entered password
# from the subroutine parameter list.
my $this_user = shift;
my $guess = shift;

# Get the user's real password record from /etc/passwd.
my (@pwdfields ) = getpwnam($this-usec);

# Use the crypt() to encrypt the user entered password
# and compare it with the encrypted password found
# in the password record in /etc/passwd.

if(crypt($guess, $pwdfields[1]) eq $pwdfields[1]) {
  # User has entered a valid password so return  a
  # non-zero value to caller routine.
  return 1;
}
# Opps! User has entered an invalid passwocd so return 0.
  return 0;
}

sub isMember{
#
# Purpose: this sub routine checks the /etc/group file to
# determine if A user belongs to a certain group
#
```

超级浏览器提醒 使用本复制品 请尊重相关知识产权!

```
# Get the username and the group name from the
# parameter list.
my $this_user = shift;
my $web_group = shift;

# Get the member list for the group from the
# /etc/group file
my ($groupName, $passwd, $gid, $memberList) = getgrnam($web_group );
# If the user is a member of this group return a nonzero
# value or else return 0.
 return 1 if ($memberList =~ /$this_user/);
 return 0;
}

sub log{
#
# Purpose: to provide a simple log
#

# Get the log entry from the parameter list
my $entry = shift;

# Append the entry in the log file.
open(FP,">>$LOG_FILE") ||die "Can't open $LOG_FILE file";
print FP $entry, "\n";
close(FP);
    }
```

该脚本分别从 USER 和 PASS 环境变量获得用户名和口令，然后它检查是否 USER 属于在/etc/group 中定义的用户组（web-user）。如果用户属于该组，脚本就检查口令是否有效。如果两个检查都通过了，脚本就返回零值；否则返回非零值。这样脚本就达到了 mod_auth_external 模块所要求的作为外部用户认证程序的条件。添加对组的检查是为了保证只有属于某个组（在/etc/group 中定义）的用户才进行认证。如果使用这一脚本，千万不要把权限很大的用户（如 root 用户）加到组中。

现在讨论如何通过下列步骤将这一脚本和 mod_auth_external 模块一起使用。

步骤 1：将 authcheckh 认证程序和关键字联系起来

在 srm.conf 文件中加入下列行：

```
AddExternalAuth myauth /usr/loacl/bin/authcheck
```

该命令告诉 Apache 将 myauth 关键字与 Perl 脚本 authcheck 联系起来。

步骤 2：建立目录访问配置

为了使用外部认证程序限制对目录的访问，在.htaccess 文件中添加下列几行：

```
AuthType Basic
AuthName Access for EveryOne
AuthExternal myauth
```

```
SetExternalAuthMethod myauth environment
require valid_user
```

AuthExternal 命令告诉 Apache 对这一目录的认证程序用 myauth 关键字标识（通过该关键字引用 authcheck 脚本）。SetExternalAuthMethod 命令告诉 Apache 使用环境变量向用 myauth 标识的认证程序传数据。最后 require 命令行告诉 Apache 允许通过认证的有效用户进行访问。

步骤 3：测试

试图访问受保护的目录将会弹出认证对话框。键入属于 web-user 用户组的有效 Unix 帐号的用户名/口令对，就允许访问该目录。

超星阅览器提醒您
使用本复制品
请尊重相关知识产权！

Tip 使用本脚本时，可按如下方式使用 tail 命令监视用户登录和登录企图：

```
tail -f /tmp/authcheck.log
```

10.3 摘 要 认 证

大家知道，基本 HTTP 认证并不安全，千万别用于敏感的应用。有一些更好的算法，但它不是服务器设计人员单独能实现的算法。浏览器开发人员也需采用新算法。这就是新算法为什么迟迟不能广泛使用的原因。可能替代基本 HTTP 认证的一种算法是摘要认证算法。摘要算法的广泛采用取决于浏览器开发人员是否将该算法加到他们的产品中。Apache 通过 mod_digest 模块支持摘要认证。

基本 HTTP 认证的问题在于口令以 UU 编码的格式通过网络发送出去，这使整个过程很不安全。在摘要认证中，口令根本不从线路上发出去，相反，网络发出的是一系列基于用户（在浏览器上）键入的口令和其他有关请求的信息而产生的数值。这些数值采用著名的 MD5 加密算法进行拼凑。最终产生的结果叫“摘要”，并通过网络发出。然后再将它与服务器上的其他项组合起来，并与服务器上储存的摘要对比。服务器根据摘要是否匹配来确定允许或拒绝用户访问。

虽然基于摘要的认证比基本 HTTP 认证更为安全，但是它要求服务器极其安全，因为在服务器上必须存储用于进行对比的摘要。

10.4 小 结

本章介绍了许多实现基本 HTTP 认证的方法，讨论了许多存储认证数据的方式，其中有文本文件、db 文件、dbm 文件以及 SQL 数据库服务器。需要牢记的重要事实是基本 HTTP 认证通过网络发送的口令是未经加密的，因此极可能被拦截。千万不要在敏感应用中采用这种方法。将来流行的浏览器将可能支持其他的认证方式，如摘要认证。大家知道，Apache 已经作好了这方面的准备。

下一章将介绍服务器状态和日志。

第 11 章　服务器状态和日志

本章内容:

- ♦监视 Apache 状态
- ♦启动日志功能
- ♦定制日志
- ♦归档日志
- ♦跟踪用户
- ♦分析日志文件
- ♦维护日志文件

超星阅览器提醒您:
使用本复制品
请尊重相关知识产权!

在阅读本章时，希望你已经运行了你的 Apache Web 站点，并且向全世界宣布了这一站点的存在。下一步该怎么办？你可曾想知道谁访问了你的可爱的站点？如果你和大多数人一样，你可能也会这样想。你可能也想知道你的 Apache 服务器在你的系统上性能怎么样。本章中将讨论怎样使用监视和日志工具来满足你的需求。

11.1　监视 Apache

Apache 允许通过 Web 监视两类信息，它们是：

- ♦服务器配置信息
- ♦服务器状态

服务器配置信息是静态的。能够快速访问正在运行的 Web 服务器配置信息是很有用的。另一方面，服务器状态信息却是动态的。使用 Apache 基于 Web 服务器的状态监视功能，能够监视诸如服务器启动时间、已服务的请求总数、总的数据传输量、子进程状态和系统资源使用情况等信息。两种类型信息都很有价值，在以下各节中将讨论这两种信息，首先讨论怎样监视配置信息。

11.1.1　访问配置信息

这项功能可由 mod_info 模块来完成，该模块对服务器配置提供了全面的描述，其中包括所有安装的模块和在配置文件中使用的命令。该模块包含在 mod_info.c 文件中。在缺省条件下，它并未编译到服务器中。可以遵循添加模块的一般过程将该模块加到配置文件中，然后运行配置脚本和 make 实用程序。mod_info 模块提供了 AddModuleInfo 命令：

语　　法：`AddModuleInfo　模块名　描述性文本`

使用范围：服务器配置，虚拟主机

本命允许在由 mod_info 模块提供的模块信息列表中添加描述性文本信息。描述性文本信息可以是任何文本格式，其中包括 HTML 文本。例如：

```
AddModuleInfo mod_auth.c 'See <A
HREF="http://www.apache.org/docs/mod/mod_auth.html">http:
//www.apache.org/docs/mod/mod_auth.htm </A>
```

这将在 mod_anth.c 列表后显示 HTML 链接，该链接提供了快速的访问方式，使它能从 Apache Web 站点获得有关该模块更多的信息。为了通过 Web 查看服务器配置信息，在 access.conf 文件中添加下列行：

```
<location /server-info>
setHandler server-info
</location>
```

超星阅览器提醒您：

可能想在 location 命令中添加<Limit>语句来限制对服务器配置信息的访问。一旦进行了配置，访问下列 URL，即可得到服务器信息：

```
http://your.host.dom/server-info
```

这将给浏览器送回有关服务器及其所有模块的完整配置信息页。

如果只需要送回服务器配置信息，使用下列 URL：

```
http:// your.host.dom /server-info?server
```

如果只需要送回单个模块配置信息，使用下列 URL：

```
http:// your.host.dom /server-info?module_name
```

如果需要快速送回所有包含的模块列表，使用下列 URL：

```
http:// your.host.dom /server-info?list
```

现在讨论怎样监视正在运行的 Apache 服务器状态。

11.1.2 允许使用状态页

mod_status 模块允许 Apache 管理员通过 Web 监视服务器。它产生服务器统计信息的 HTML 页，同时也生成机器可读（本章稍后将讨论）的页，两个页中包含的信息如下：

- 服务器系统的当前时间
- 服务器最近一次重启时间
- 服务器启动后的运行时间
- 到目前为止服务的访问总数
- 到目前为止传输的字节总数
- 服务请求的子进程数
- 空闲子进程数
- 每个子进程状态、子进程服务的请求数以及该子进程传输的字节总数
- 每秒平均请求数、每秒传输的字节数、每次请求平均传输字节数
- 目前每个子进程 CPU 占用率及 Apache 的总的 CPU 占用率
- 当前主机及处理的请求

Note

如果状态模块所提供的 Web 页中没有列出所有上述的字段，应为配置文件中的 AUX_CFLAGS 命令添加-DSTATUS 选项，然后重新配置并编译 Apache。需要注意的是在某些系统上这样做可能会给系统性能带来一点损失。

在缺省的情况下，本模块未被编译在 Apache 标准版中。因此，应将该模块加到配置文件中，然后运行配置脚本和 make 实用程序生成新的 Apache 可执行文件。用 httpd -l 检查 mod_status 模块是否包含在可执行文件中。

11.1.2.1　查看状态页

当编译了 mod_status 模块并将它包括在 Apache 服务器中以后，需要定义一个 URL 位置，Apache 将用它来显示信息。换言之，需要告诉 Apache 哪个 URL 将会指向服务器统计数据。假设机器域为 yourdomain.com，并想用下列的 URL：

```
www.mydomain.com/apache-status
```

则可使用<Location...>容器告诉服务器想使用 mod_status 模块中的 server-status 处理程序来处理这个 URL，下列代码可实现此项任务：

```
<Location /apache-status>
SetHandler server-status
</Location>
```

其中 SetHandler 设置了前述 URL 的处理程序（Server_status），典型情况下这段配置信息应放在 access.conf 文件中，但它实际上可能在三个配置文件的任何一个中。一旦在三个配置文件的任何一个中加入了上面的配置信息，就可以重启服务器，并从浏览器访问前面的 URL。图 11-1 显示了状态页的示例。

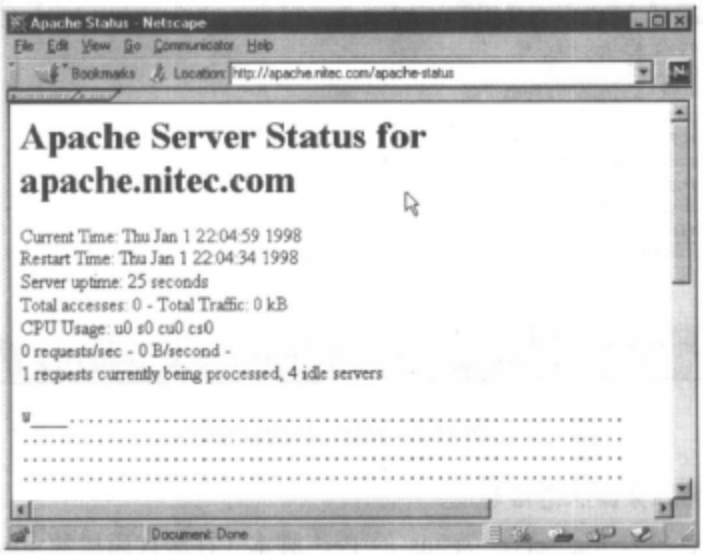


图 11-1　Apache 状态页示例

前面使用的<Location..>容器允许任何人使用这个 URL 查看服务器状态。当越来越关注安全的时候这样做恐怕未必是什么好事。为了确保只有在域内的机器才可以访问状态

页，可用下面的配置代替前面配置：

```
<Location /apache-status>
setHandler server-status
order deny, allow
deny from all
allow from .yourdomain.com
</Location>
```

其中 yourdomain.com 应该用自己的域名代替。如果需要选择一个或几个主机可以访问状态页，只需简单地将主机名列在 allow 命令行后面。

如果浏览器支持自动刷新，则可以让状态页自动更新。访问 Web 页 http: //www.yourdomain.com/server-status?refresh=N，可使状态页每 N 次刷新一次。

11.1.2.2 简化状态显示

由该模块所显示的状态页提供了额外的信息，因此它并不适合作为数据文件供其他分析数据的程序使用。例如，想使用电子表格程序根据服务器状态数据制作一幅图，就必须手工清除无用数据。但该模块提供了从同一 URL 生成机器可读的输出结果的方法。

为了简化状态显示，在 URL 后面加上“?auto”，该查询字符串告诉 Apache 显示简化的输出，如图 11-2 所示。

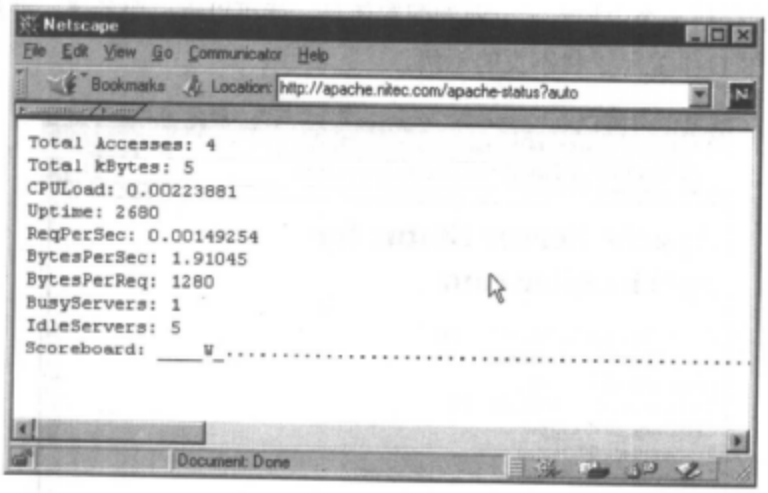


图 11-2 机器可读的状态输出

11.1.2.3 存储服务器状态信息

Apache 还带有名为 log_server_status 的 Perl 脚本（在源代码树的 support 目录中），该脚本能定期地将服务器状态信息（使用?auto）存储在文本文件中。

可以将该脚本作为 cron 作业来运行，按需要的时间间隔收集状态信息。使用该脚本前，必须编辑脚本以修改下列配置变量的值：

```
$wherelog
```

```
$port
$server
$request
```

将$wherelog 的值改为存储由脚本生成的文件的路径。$port 变量的值应是将要监视的服务器的端口号。如果服务器运行在标准的 HTTP 端口上，使用缺省值 80 就行了。$Server 变量应该用服务器的主机名赋值。如果这一脚本和服务器运行在同一系统上，用缺省值 localhost 就可以了。如果服务器运行在另一台机器上，$Server 变量的值必须是完整的主机名（如 www.mydomain.com）。$request 变量的值应设成在<Location..>命令中指定的内容再加“?auto”查询字符串。

如果不喜欢脚本使用的记录格式，可修改下列行来满足特殊要求：

```
print OUT "$time:$requests:$idle:$number:$cpu\n";
```

这一脚本使用 socket 连接到 Apache 服务器并发出 URL 请求，因此必须确保 Perl 支持 socket。

11.2　建立日志文件

知道 Web 服务器的状态和配置信息对管理服务器是很有帮助的。但是，知道谁在访问 Web 站点也很重要，并且令人兴奋。可以用 Apeche 服务器记录日志的特征来获得这些信息。在下面的几节中，将详细讨论如何记录日志，以及如何最大限度地利用 Apache 的记录日志模块。

当市场上出现 Web 服务器软件的时候，同时开始出现许多 Web 服务器日志分析程序。这些程序成为 Web 管理员日常工作的一部分。随之而来的是这些日志文件不兼容，这使日志分析成为相当困难和繁重的工作。单独一个日志分析程序并不能适用于各种类型日志。随后出现了通用日志格式（CLF）规范，这使所有 Web 服务器以基本上相似的方式书写日志文件，因此从一个服务器到另一个服务器的日志分析就变得简单多了。

Apache 标准版包括名为 mod_log_config 的模块，并用它来记录日志。在缺省情况下，它写 CLF 日志文件，但可用 LogFormat 命令来改变这一设置。在大多数情况下，CLF 文件能够满足记录要求。CLF 文件中的每一行的内容解释如下。

在 CLF 文件内对每个请求均有一单独行，该行由用空格分开的几个标识符组成：

```
host ident anthuser date request status bytes
```

如果标识符无值则以减号 “-” 表示。标识符的含义如下：

♦Host——客户端主机的全称域名或 IP 地址。

♦Ident——如果允许使用 IndentityCheck 命令，并且客户机运行了 identd，那么它存放客户端报告的识别信息。

♦Authuser——如果所请求的 URL 要求成功的基本 HTTP 验证，那么它的值应为用户名。

```
TransferLog logs/access.log
```

当参数是指向外部程序的管道时，日志信息发送到外部程序的标准输入（STDIN）中。

注意，如果一个新程序从主服务器继承了 TransferLog，那么它就不是为虚拟主机启动的。如果使用了程序，那么它由启动 httpd 的用户所拥有。如果用 root 帐号启动服务器，那么它就由 root 所拥有。要保证这一程序是安全的。

超星阅览器版本
使用本复制品
请尊重相关知识产权！

11.2.2　LogFormat 命令

语　　法：`LogFormat format [nickname]`

缺 省 值：`LogFormat "%h %l %u %t \"%r\" %s %b"`

使用范围：服务器配置，虚拟主机

该命令设置 TransferLog 命令命名的缺省日志文件的格式。如果在命令行中包括了格式的别名（nickname），那么可在其他 LogFormat 或 CustomLog 命令中使用它，而无需重复整个格式字符串。定义别名的 LogFormat 命令不干其他事情，只定义别名，而并不实际应用这一格式。

参照本章后面的“定制日志文件”一节，该节详尽讨论了可用的格式选项。

11.2.3　CustomLog 命令

语　　法：`CustomLog file-pipe format-or-nickname`

使用范围：服务器配置，虚拟主机

和 TransferLog 命令类似，本命令能够向日志文件或外部程序发送日志信息，同 TramsferLog 命令不同的是它允许以参数的形式指定日志格式。

该格式参数为日志文件的每行指定格式，格式选项和 LogFormat 命令的参数完全相同。如果格式中有空格，那么应该用双引号括起来（这种情况几乎难以避免）。

也可以用一个 LogFormat 命令定义的格式的别名来代替实际的格式串。

别名只能在 Apache 1.3 及更新版本中使用。另外也要注意 TransferLog 和 CustomLog 命令在每一服务器中均可使用多次，以便将单个请求记录在多个日志文件中。

11.2.4　CookieLog 命令

语　　法：`CookieLog filename`

使用范围：服务器配置，虚拟主机

CookieLog 命令用来设置 cookie 日志的文件名，文件路径是相对于 ServerRoot 的。本命令为了与 mod_cookie 模块兼容才保留下来，因此并不推荐使用此命令，可用用户跟踪模块的相应命令代替。本章稍后将讨论用户跟踪模块 mod_usertrack。

11.3 定制日志文件

超星阅览器提醒您：

虽然 CLF 格式可以满足大多数日志要求，但有时能够定制日志数据也是有很用的。例如，可能想将访问你的站点的浏览器类型写入日志，这样 Web 设计小组就可据此决定避免或使用哪种浏览器专用 HTML。或者你想知道哪个站点将访问者引入到你的站点（即引用了你的站点）。所有这些在 Apache 内能够轻松实现。缺省日志模块 mod_log_config 支持定制日志。

定制格式可以用这一模块的 LogFormat 或 CustomLog 命令来设置，这两个命令的格式参数是字符串，它既可以包含任意文字字符，也可以有特殊的“%”格式指定符。当串内有文字字符时，每次请求都将它们复制到日志文件中，然而“%”格式指定符用相应值来替换，这些特定的%指定符见表 11-2。

表 11-2 特定的%指定符

%b	除 HTTP 头标外发出的字节数
%f	请求的文件名
%{variable}e	环境变量 VARIABLE 的内容
%h	发请求的远程主机
%{IncomingHeader}i	输入头标的内容，即发到服务器的 HTTP 请求的头标行。末尾的字符 i 表示这是来自客户端的头标（输入头标）
%I	如果允许使用 IndentityCheck 命令，并且客户端运行 identd，那么它是客户端报告的识别信息
%{Foobar}n	另一模块发来的 note Foobar 的内容
%{OutgoingHeader}o	输出头标的内容，即应答的头标行，字符 o 表明它是服务器（输出）的头标
%p	对请求作出应答的端口
%p	对请求作出应答的进程 ID
%r	请求的首行
%s	服务器响应请求返回的状态。当请求重定向后注意该格式指定符的值仍是最初的请求状态。如果想存储重定向请求状态用%..>s 代替
%t	请求的时间，时间格式和 CLF 的格式相同
%{format}t	时间，其格式由 Format 参数给出（格式细节可参照第 7 章）
%t	响应该请求服务器所花时间，以秒计

续表 11-2

%u	当所请求的 URL 成功地通过基本 HTTP 验证时，该格式指定符的值是用户名。如果服务器在验证后发回 401 状态，那么该值可能是假用户名
%u	所请求的 URL 路径
&v	请求发向的服务器名或虚拟主机

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

在前面的每个格式指定符中可以包含条件信息，条件可以出现（或不出现）某个 HTTP 状态码。举个例子，假如想记录将用户指向不存在的页的所有引用的 URL。这种情况下，服务器产生 404 状态（未发现）头标。那么可使用格式指定符将引用的 URL 写入日志：

```
'% 404 {Referer} i'
```

类似的，为了将导致不正常状态的引用的 URL 写入日志，可这么做：

```
'%! 200, 304, 302 {Referer}i'
```

注意“！”字符的使用，它表明当所列出的服务器状态不出现时条件成立。

类似的，为了在 CLF 格式指定符的后面添加额外信息，可以扩展 CLF 格式，该格式是由下面的格式串定义的：

```
"%h %l %u %t \"%r\" %s %b"
```

例如：

```
"%h %l %u %t \"%r\" %s %b \"%{Referer}i\" \"%{User-agent}i\""
```

本格式说明将 CLF 格式数据写入日志，并且在每个日志项中添加在客户提供的头标中发现的 Referer 和 User-agent 信息。

现已经学会了如何在日志文件中添加非 CLF 数据，但是怎样才能将这些数据存入多个日志文件中呢？下一节讨论怎样将 CLF 数据和非 CLF 数据存入多个日志文件。

11.4　建立多个日志文件

有时很有必要建立多个日志文件。例如，正使用一个不能处理非 CLF 数据的日志分析程序，可能会想将非 CLF 数据写入另一个文件中。用 TransferLog 或/和 CustomLog 命令可以轻松地建立多个日志文件。简单地重复这两个命令就能建立多于一个日志文件。

例如，想生成一个标准的 CLF 访问文件和一个含有所有引用的 URL 信息的定制日志文件，可以像下面这样做：

```
TransferLog logs/access_log
CustomLog logs/referer_log  "%{Referer}i"
```

当在主服务器配置中定义了 TransferLog 或 CustomLog，并且定义了一个虚拟主机，与虚拟主机有关的日志记录也记录在这些日志中。例如：

```
TransferLog logs/access_log
CustomLog logs/agents_log  "%{User-agent}i"
```

```
<Virtual Host 206.171.50.51>
ServerName reboot.nitec.com
 DocumentRoot  /www/reboot/public/htdocs
 ScriptAlies /cgi-bin/  /www/reboot/public/cgi-bin/
 </VirtualHost>
```

上面虚拟主机 reboot.nitec.com 的<VirtualHost..>容器标记中并没有定义 TransferLog 和 CustomLog 命令。所有日志信息将存在 log/access_log 和 logs/agentslog 文件中。现在如将下列行加入到<VirtualHost...>容器内：

```
TransferLog vhost_logs/Reboot_access_log
```

那么虚拟主机 reboot.nitec.com 的所有日志信息将存在 vhost_logs/reboot_access_log 文件中。log/access_log 和 logs/agents_log 都不用于 reboot.nitel.com 虚拟主机。

11.5 存储特殊信息

本节介绍几个能提供缺省日志模块（mod_log_config）也可提供的功能的几个特殊日志模块。提供此信息是为了完整和向后兼容的缘故。

11.5.1 存储用户代理信息

Apache 提供了另一个名为 mod_log_agent 的模块，它允许在单独的日志文件中存储用户代理（Web 浏览器，机器人）的信息。该模块未编译到 Apache 标准版中，因此需要像通常那样把它加到 Apache 可执行文件中。该模块只有一个 AgentLog 命令：

语　　法：AgentLog filename| "|/path/to/external/program"

缺 省 值：AgentLog logs/agent_log

使用范围：服务器配置，虚拟主机

该命令告诉 Apache 服务器将用户请求中的 UserAgent 头标发送到指定的文件或程序。和 TransferLog 命令类似，可以指定文件名，也可以指定一个程序的完整路径名作为参数。

有一点需要注意：使用 CustomLog 命令，加上%{user-agent}i 格式指定符，无需使用本模块就可将 UserAgent 头标中的信息写入日志。换言之，该模块并没有提供标准版的 mod_log_config 模块所没有的功能。如果本模块能够忽略所在网上的本地访问的信息，那么这些数据将会更有趣。

接着来看看能在日志文件中存储引用者信息的模块。

11.5.2 存储引用者信息

知道谁引用了你的 Web 站点是很有帮助的，因为它让你知道在 Web 上谁是你的朋友。它也有益于你决定将广告费花在哪儿。

能够分辨出谁向你的 Web 站点转发了请求，这对建立与其他 Web 站点的关系是极有用的。如果从某个站点收到许多引用，则可能也想对它表示谢意，并提供从你的站点到它的一个链接。但是，要记住先得到他们的允许。另一方面，如果从你不关心的或不需要发

生关系的 Web 站点收到一些引用，则可以从日志中找出这些站点，然后礼貌地要求它们删掉对你的站点的引用。

超星阅览器提醒您：

mod_log_referer 模块提供了将请求中的 Referer 头标写入日志的功能。当然，标准的 mod_log_config 模块的 CustomLog 命令使用%{Referer}I 格式指定符也能完成同样功能。那么，使用本模块的好处何在？该模块和 mod_log_config 有些不同，它允许忽略某些主机。通过这种方式可以仅将真正用户的信息写入日志，而本地主机的引用就忽略掉了。

需要自己将该模块编译到 Apache 可执行文件中，它有以下命令。

11.5.2.1　RefererLog 命令

语　　法：`RefererLog filename | "|/path/to/external/program"`

缺 省 值：`RefererLog logs/referer_log`

使用范围：服务器配置，虚拟主机

本命令设置接收输入请求的 Referer 头标的日志文件名或程序名。

关于路径限制和如何使用外部程序的详细情况请参考本章的 TransferLog 命令。

11.5.2.2　RefererIgnore 命令

语　　法：`RefererIgnore string string ...`

使用范围：服务器配置，虚拟主机

RefererIgnore 命令添加在 Referer 头标应该忽略的字符串列表。如果列表中任何一个字符串出现在某个 Referer 头标中，那么该请求的引用者信息将不写入日志，例如：

```
RefererIngore yourcompany.com
```

该命令即可避免将 yourcompany.com 域中主机的引用写入日志。如果 yourcompany.com 是内部网的域名，就能够忽略所有来自内部网络上的引用。

日志文件的格式相当简单，每次引用占一行。每行的格式如下：

```
uri -> document
```

其中 uri 是引用了客户端正访问的文档的 URI，对于被引用文档来说该文档是本地 URL。

11.5.3　存储 cookie

到现在为止，已经讨论了几个日志选项，但所有选项中无一能唯一地标识访问者。这一点很重要，因为如果知道哪个请求是哪个访问者发出的，就能更好地理解你的 Web 的内容（content）如何被使用。例如，在你的 Web 站点的某处有一个相当酷的网页，并且你有办法在日志中标出访问者。如果查看日志文件，并发现有许多访问者从一个网页到另一个网页地浏览最后找到这个很酷的网页，那么就可能会考虑重新设计站点结构，使访问者很快就找到这个网页。Apache 有一个名为 mod_usertrack 的模块，它通过将 HTTP cookie 写入日志文件，可跟踪 Web 站点的访问者。

HTTP Cookie：

HTTP Cookie 并不是用甜饼面团做的，它只是服务器给予 Web 浏览器的一段信息。该信息以 key=value 对的形式存储，并能够与整个 Web 站点或站点上某个特定的 URL 联

本命令设置允许或禁止 cookie 的自动生成功能。设置为“on”时，Apache 为所有新请求发送用于跟踪用户的 cookie，图 11-3 显示了该类 cookie 的例子。

图 11-3　Apache 向浏览器发送用于用户跟踪的 cookie

CookieTracking 命令能够以单个服务器或单个目录为基础将该功能设为“on”或“off”。在缺省情况下，编译 mod_usertrack 模块并不激活 cookie。

11.6　使用错误日志

本章中曾讨论过将各种有趣的信息写入日志的方法，然而，在你的站点上必须使用的记录功能是错误日志。如果没有错误日志你就无法找出出了什么问题和错误出现在哪里。

有趣的是，错误记录功能是由 Apache 内核支持的。ErrorLog 命令能够将 Apache 遇到的各种类型错误均写入日志。

Cross-Reference　更详细的内容请查阅第 4 章中有关本命令的描述。

本节将讨论如何将 Apache 错误日志包含在大多数 Unix 系统中广泛使用的 SysLog 功能中。

SysLog 是（服务器）守护进程发送日志信息的传统方式。你可能会问：“Apache 是一个守护进程，那它为什么不能写 SysLog 呢？”。实际上，这是可以的。你所需要做的只是将配置文件中已有的 ErrorLog 命令用下面的替换：

```
ErrorLog SysLog
```

然后重启 Apache。用 Web 浏览器访问你的 Web 服务器上不存在的网页，然后查看 SysLog 文件中是否有一个 httpd 项。应看一下/etc/syslog.conf 文件，其中有 httpd 消息出现在何处的提示。

例如，列表 11-1 显示了 Linux 系统的/etc/syslog.conf 文件。

列表 11-1：　/etc/syslog.conf

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
# kern.*          /dev/console

# Log anything (except mail) of level info or higher.
```

```
# Don't log private authentication messages!
*.info;mail.none;authpriv.none          /var/Log/messages

# The authpriv file has restricted access.
authpriv.*              /var/Log/secure

# Log all the mail messages in one place.
mail.*        /var/Log/maillog

# Everybody gets emergency messages, plus log them on another
# machine.
*.emerg                 *
# Save mail and news errors of level err and higher in a
# special file.
uucp,news.crit              /var/Log/spooler
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

列表中有两行很重要（当考虑 Apache 时），它们是：

```
*.info; mail.none; authpriv.none  /var/log/messages
*.emerg            *
```

第一行告诉 syslog 将所有 info 类型的消息（除 mail 和个人验证外）均写到/var/log/messages 文件中，第二行说明所有紧急消息应写到所有日志文件中。使用 LogLevel 命令可以指定 Apache 将哪类消息发给 syslog。例如：

```
ErrorLog syslog
LogLevel debug
```

上例中，通知 Apache 将 debug 信息发到 syslog。假如想通过 syslog 将调试信息存到另一个文件中，那么你需要修改/etc/syslog.conf 文件，例如：

```
*.debug  /var/log/debug
```

在/etc/syslog.conf 中加上这一行后，重启 syslogd（kill -HUP syslogd PID）。这样 Apache 就会允许将 debug 信息存到/var/log/debug 文件中。有几个记录级别设置：

- Emerg——紧急消息
- Alert——警戒消息
- Crit——关键消息
- Error——错误消息
- Warn——警告消息
- Notic——注意
- Info——信息
- Debug——调试级记录的消息，包括产生消息的源文件名及其行号，该消息有助于程序的开发和调试。

Tip

如果想查看 syslog 文件或其他日志文件的最新更改的内容，可以使用大多数 Unix 系统都有的 tail 实用程序。例如，如果想看名为/var/log/httpd_errors 的日志的最新内容，请用：

```
tail -f /var/log/httpd_errors
```

11.7　分析日志文件

到目前为止，已经掌握了如何建立以 CLF 格式为基础的日志和定制日志。现在应该找一种分析这些日志的方法以利用这些记录下来的数据。每个人的日志分析需求各不相同，有时需产生大量的报表，有时也许只想对日志作简单的检查。对于较简单的任务，最好使用手边现成的工具，大多数的 Unix 系统都有大量实用程序和写脚本的工具来完在这项工作。

使用本复制品请尊重相关知识产权！

来看一下如何用 Unix 实用程序得到所有的主机名列表。如果使用了具有 CLF 支持的定制日志或缺省记录功能，就可以轻松地得到主机名列表。例如：

```
cat  /path/to/httpd/access_log | awk '{print $1}'
```

该命令打印出所有主机的 IP 地址（如果允许使用 DNS 查找功能，将列出主机别名的清单）。cat 实用程序列出 access_log 文件的内容，并将输出结果通过管道送给 awk 解释程序。awk 解释程序使用打印语句打印出每行的第一个字段。该命令将打印出所有主机名。如果想将你所在网络上的主机排除在外，这种情况下应这样做：

```
cat /path/to/httpd/access_log | awk '{print $1}' | egrep -v
'(^206.171.50)'
```

其中 206.171.50 应该用自己的网络地址代替。注意，在此处假设你的网是 C 类网。如果你的网是 B 类，只需写 IP 地址的前两个 8 位字节。这种方式允许用 egrep 实用程序将自己的主机排除在外，通过-V 选项告诉该程序只显示不以 206.171.50 网络地址开头的主机。仅仅这样做可能还不会令人满意，因为项与项之间可能有重复。因此实现该功能的最好的形式应是：

```
cat /path/to/httpd/access_log | awk'{print $1}'  | Uniq | egrep
-v '(^206.171.50)'
```

在此行命令中，uniq 实用程序过滤掉重复出现的主机名。因此，在显示的主机列表中，每个主机只占一项。当然，如果想知道访问你的 Web 站点的主机的总数，可将最终结果通过管道送给带-l 选项的 wc 实用程序：

```
cat /path/to/httpd/access_log | awk'{print $1}'
     | Uniq | egrep -v '(^206.171.50)' | wc -l
```

该命令给出总的行数（即所有主机的数目）计数。

使用 Unix 实用程序，能够迅速地获得所需的信息。然而这种方法却要求具备一些 Unix 知识，并且并不总是很方便，因为你的老板可能想要一个漂亮的报表而不是一大堆枯燥的文本列表。这种情况下，可以开发自己的分析程序，也可以使用第三方提供的分析软件。

有许多第三方提供的 Web 服务器日志分析工具。大多数工具需要日志文件具有 CLF 格式，因此要保证你的日志使用的是 CLF 格式。表 11-3 列出了一些这样工具及其网址。

找出哪种工具适合使用的最好办法是试一下。或者至少访问它们的站点并比较它们的功能。所发现的两个非常有用的实用程序是 Wusage 和 wwwstat。

表 11-3 第三方提供的日志分析工具

产品名	产品 URL
WebTrends	www.webtrends.com/
Wusage	www.boutell.com/wusage/
wwwstat	www.ics.uci.edu/pub/websoft/wwwstat/
http-analyze	www.netstore.de/Supply/http-analyze/
pwebstats	www.unimelb.edu.au/pwebstats.htm/
WebStat Explorer	www.webstat.com/
AccessWatch	http://netpressence.com/accesswatch/

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

Wusage 是令人喜爱的商用日志分析软件。它的可配置性极好，并能用该公司的著名的 GD 图形库生成大量图形报表。Wusage 以二进制形式发布。Wusage 的评价版对许多 Unix 和 Windows 平台均是免费的。

wwwstat 是自由发布的分析程序。它是用 Perl 写的，因此必须在运行本应用程序的系统上安装 Perl。wwwstat 输出的综合结果可以由 gwstat 读取以用来生成有趣的图形。

现在可能意识到在 Apache 中生成日志很简单，也很有用。这样做可更多地知道 Apache 服务器能做什么。日志能帮助检测问题，认识站点的问题，找出站点最好的特征等。难道只有好处没有缺点吗？ 如果回答不是的，那么就猜对了。日志文件占用大量的宝贵的磁盘空间，因此必须定期进行维护。

11.8 日 志 维 护

使用日志可能会节省很多工作时间，但是日志本身确实增加了一些额外的工作：日志需要维护。对于访问频繁或有许多虚拟域的 Apache 站点讲，日志文件可能会在短期内变得很大。这可能会造成磁盘故障。在日志文件太大时，应该转储日志。

转储日志有两个选择：使用和 Apache 一起提供的 rotatelog 实用程序，或是使用在多数 Linux 系统上都有的实用程序 logrotate。

11.8.1 使用 rotatelog

Apache 带有称为 rotatelog 的实用程序，你可以这样使用该程序：

```
TransferLog "|/path/to/rotatelogs <logfile> <rotation
time in seconds>"
```

例如，若想将访问日志每隔 86400 秒（即 24 小时）转储一次，可使用下行命令：

```
TransferLog "|/path/to/rotatelog /var/logs/httpd 86400"
```

这样就将每天的访问日志信息存储在名为/var/logs/httpd.nnnn 的文件中，其中 nnnn 代表一个大的数。

11.8.2　使用 logrotate

logrotate 实用程序能转储、压缩日志文件并将其作为邮件发出去。它的设计目标是方便日志管理。它能够以日、周、月或日志文件大小为基础自动转储、压缩、删除以及邮递日志文件。通常 logrotate 是作为日常的 cron 作业运行的。阅读 logrotate 手册以获得更多信息。

如果你的系统支持 logrotate 功能，则配置应该和/etc/logrotate.conf 文件中所列的类似：

```
/path/to/httpd/access_log {
compress
rotate 5
mail {hyperlink mailto:webmaster@yourdomain.com}
errors {hyperlink mailto:webmaster@yourdomain.com}
size =1024K
postrotate
  kill -HUP  'cat /path/to/httpd.pid'
endscript
}
```

该配置指定当日志文件/path/to/httpd/access_log 超过 1M 时，旧的日志文件就被压缩并经五次转储后发到你的域中邮件服务器的 Webmaster 帐号的邮箱中，而不是删除掉。处理日志文件过程中出现的任何错误都将发给 root@yourdomain.com。

11.9　小　　结

在本章中，学习了如何监视 Apache 服务器和将信息记入日志文件、怎样快速访问服务器的配置、怎样监视正在运行的 Apache 服务器的状态，以及如何用 CLF 格式和定制格式建立日志文件，也学会了怎样用第三方提供应用程序分析日志文件。监视和记录日志提供了大量数据，这些数据对于系统管理员顺利管理 Web 服务器相当重要，它们也对你的 Web 站点的市场营销颇有帮助。

下一章将讨论 Web 服务器的安全。

第 12 章　Web 服务器的安全

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

本章内容：

- 理解 Web 的安全
- 掌握总的的安全模型
- 认识潜在的 CGI 和 SSI 安全风险
- 将 CGI 和 SSI 的安全风险降到最低
- 保护 Web 的内容
- 提出安全管理的策略

曾经因为担心你的 Web 服务器的安全而不能入睡吗？假如是这样的话，那你并不是唯一的——许多 Web 管理员也是这样。许多 Web 管理员从局域网（LAN）管理员开始做起。几年前，大多数公司并不理解 Internet。那时，公司的局域网是孤立的，运行的协议并不会将数据包发送到 Internet 上。

最近几年，这种状况发生了急剧变化。大多数公司现在考虑将其网络直接连接到 Internet 上作为其网络设计的一部分。将局域网连到 Internet 上变得相当简单，但是许多局域网管理员并没有掌握 Internet 的内部工作机制，因此他们感到对安全问题并不能够得心应手地应付，甚至根本没有意识到安全风险！

遗憾的是，对 Internet 安全的详细讨论超出了本书的范围。在本章中，将对讨论中的安全模型进行概述，并详细讨论它会对 Web 服务产生什么样的影响。你将了解哪里潜伏着安全风险，以及如何采取预防措施，以减少新的风险。

12.1　理解 Web 的安全需求

安全需求对新手来说往往并不清楚。许多管理员只在有出现安全缺口后才关心安全问题。很难让管理员对一无所知的事担心，因此在此列出几个隐藏的风险。

当将 Web 服务器运行起来并使它能为世界上其他人服务时，同时也为其他人进入你的网络打开了窗口。你确实打算这样做吗？绝大多数人将使用你的 Web 站点所提供的内容，但是有些人却可能从你的窗口中找漏洞来获得他们本不应该得到的信息。这些人中有的是想把形势搞成一团糟的破坏主义者，也有的是信息窃贼。无论哪种情况，只要有人成功地找到了漏洞，你就可能发现你的 Web 站点已被篡改成猥亵的内容，甚至可能丢失机密数据。有时攻击并没有直接影响你的站点。但渗入者可能会利用你的服务器或其他资源进入其他网络，并使你面临法律的风险。这些情况没有一种是所期望的。

为了避免发生这种风险和令人难堪的事情，必须保证已为你的 Web 服务制订了安全

要求。本章中假设你的 Web 站点有下列安全要求：

♦维持 Web 上发布的信息的完整性。

♦防止使用 Web 服务器主机作为侵入你的网络的跳板（这可能导致泄密以及信息资源可用性和完整性的破坏）。

♦防止使用 Web 服务器主机作为侵入其他网络的跳板（这可能导致你的组织承担破坏的责任）。

大多数网络安全事故发生的原因是软件配置不当而造成的机密信息易于受到攻击。软件可能是 Web 服务器本身，或者是 Web 服务器运行的应用程序（如 CGI 程序、SSI 和服务器 API 程序）。这会导致无意的机密信息泄漏，然后攻击者就会用这些机密信息发动攻击。

超星阅览器提示：
使用本复制品
请注意知识产权！

能够保证你的 Web 服务器配置牢不可破并且它所运行的应用程序也相当安全吗？不能，因为对 Web 服务器的攻击可能来自不同的方面。攻击者可能通过与 Web 毫不相关的应用程序，或通过操作系统的隐错，甚至通过糟糕的网络结构攻击你的 Web 服务器。

为了保证安全必须要检查每一个细节问题，这可能是不能实现的任务。但请记住：不存在绝对安全的环境，不存在对所有安全问题的简单的解决方案，因为我们并不知道所有的安全问题。你的任务是每时每刻都尽可能地提高系统的安全。

12.2　安全检查点

为了提高公用 Web 服务器的安全，应关注几个安全检查点。图 12-1 显示了总的安全检查点示意图，该图说明在权衡连接 Internet 的网络安全必须仔细检查的安全检查点。

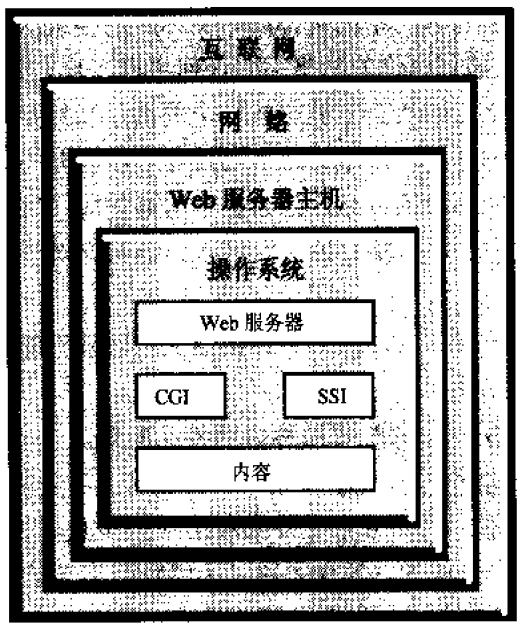


图 12-1　安全检查点

12.2.1 你的网络

要考虑的第一个检查点是你的网络和它是怎样连到 Internet 上的。真正的问题是你把 Web 服务器放在哪里。如果你的网络配置和图 12-2 一样，那么你的网络基本上未受到任何保护。

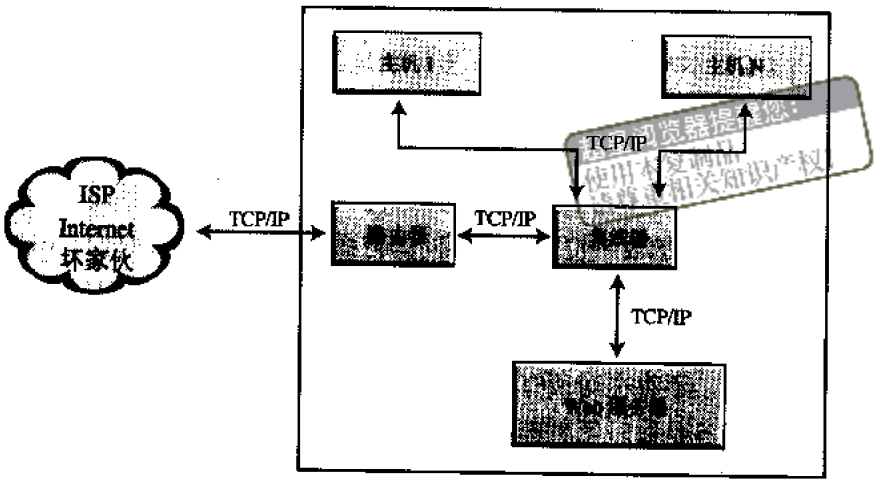


图 12-2 连接 Internet 的简单网络

为了简便起见，假设到 Internet 只有一条连接线路。

在这个网络模型中，所有的网络流量都路由到你的网络或从你的网络路由出去，换言之，Internet 上任何人都能够到达你的网络上的任何主机，反之亦然。这也意味着 Internet 上的黑客一定会从你的网络上的所有主机中找一个弱点。当然并不是所有的主机上都运行着为他人服务的服务器。但是，即使你的网络上只有一台能提供文件共享服务的 Windows 主机，它也有可能成为潜在的攻击目标。因为整个网络使用 TCP/IP 协议，Windows 工作站上的一个简单的文件共享选项可能导致黑客进入，该网络对于想闯进的犯罪分子敞开了大门。

如果你的网络是这种配置的话，查一下路由器文档看能不能实施基于 IP 的限制。建议不惜代价极力避免这种配置。

下一种配置如图 12-3 所示，该配置被称为牺牲品配置。

Web 服务器放在防火墙和内部局域网之外，局域网上的主机通过防火墙收发 Internet 数据包。通过这种方式内部主机即可免受外部攻击（至少从理论上是这样）。然而 Web 服务器对外部攻击却是完全开放的，这就是称其为牺牲品配置的原因。在这种配置中：

♦入侵者不能观察或获取内部主机间的网络信息。这些信息流包括认证信息、专用商务信息、个人数据以及许多其他敏感数据。

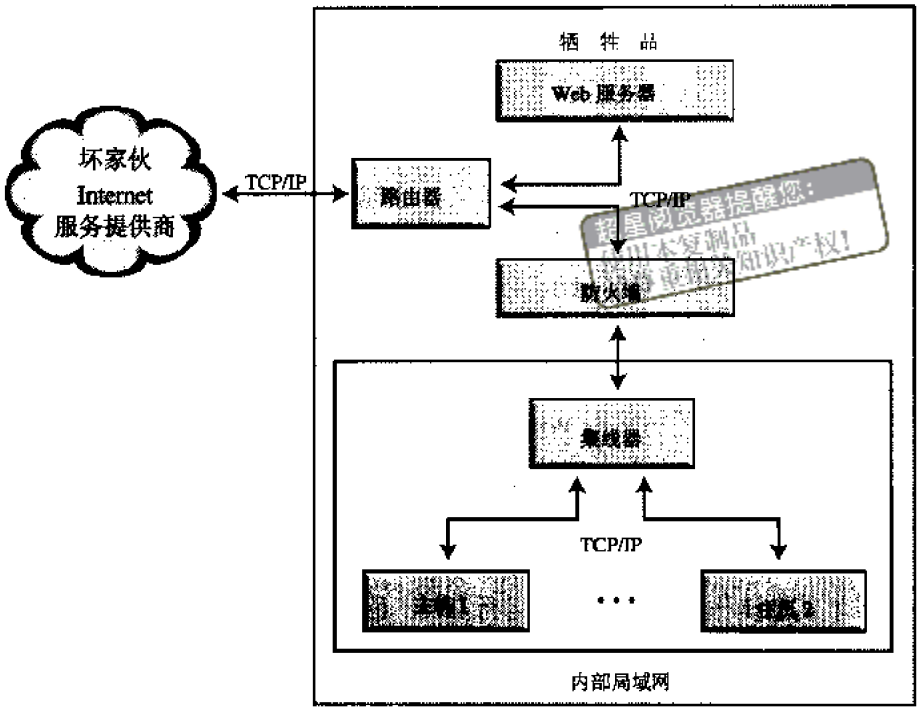


图 12-3　牺牲品网络配置

♦入侵者不能到达内部主机，或获得内部主机的详细信息。

牺牲品配置可以抵御这两种威胁，并将 Web 服务器与内部网及其上的信息流隔离开。

Tip　在牺牲品配置中，关掉路由器的源路由是明智的。这样 Web 服务器就不能用来向内部网上的主机转发数据包了。

某些人喜欢将 Web 服务器也放到防火墙后。在这种配置中，防火墙成了内部局域网和 Web 信息流的网关，并且防火墙配置比较复杂。这种复杂性会导致防火墙的安全漏洞，反而不能达到目的。

一般的想法是隔离内部网，尽量不让本地网络之外的人窥视它。一定得考虑在 Internet 到 LAN 的路径上装防火墙。如果不能选用防火墙则可以考虑代理服务器。代理服务器无需将两个网络连在一起即能够将一个网络的请求发到另一个网络，这可能是个好办法，因为它也提供了极优秀的记录日志的功能。

正如所看到的，你应该更关注你的内部网络而不是 Web 服务器本身。因为将 Internet 可访问的 Web 服务器引入网络，网络就更可能受到利用 Web 服务器或运行在其上的应用程序（如 CGI 程序）的安全漏洞而进行的攻击。选择一个将局域网从 Web 服务器中隔离的配置方案可以将这种攻击的可能性大大减少。

虽然应该将 Web 服务器放到网络的安全部分之外，但是还是应该尽量避免让黑客攻击它。

下一节中讨论提高操作系统和 Web 服务器安全的方法，然后将讨论用来增强服务器

安全的各种措施。

12.2.2 操作系统

Apache 几乎能够在所有 Unix 系统平台上运行，并且已经开始运行在 Windows NT、Windows 95、甚至新的 Macintosh 操作系统上。不能仅因为它能够运行在任何平台上就采用任意的操作系统来运行 Apache，选择合适的操作系统是系统安全的关键因素。著名的 Unix 以及 Unix 类操作系统，如 BSDI、FreeBSD、SunOS、Solaris、HP-UX、Digital Unix 以及 Linux 在 Internet 管理员中都很受欢迎。虽然这些操作系统的质量和可靠性有所不同，但它们都是可用作服务器的操作系统。

超星阅览器版本 使用本复制品 请尊重知识产权!

如果不买 Unix，则可能想采用 Windows NT 服务器平台。Windows NT 是 Unix 的有力竞争对手，未来可能占领很大一部分 Unix 的市场份额，然而对于 Internet 来说它还太年轻。大多数的 Unix 及 Unix 类的操作系统长期以来承受了大量的侵入和攻击，它们成为幸存者并且依然——虽然不能毫无瑕疵——高高矗立。

一旦选择了某个操作系统，则应在运行 Web 服务器的操作系统上去掉任何其他不打算用的功能。例如不需要 SMTP/POP 邮件或 FTP 服务，那么应禁止该项服务或彻底从系统中删除邮件服务器程序。如果不需要大量功能强大的系统实用工具，那么就把它们删掉。确保你的 Web 服务器仅是一个 Web 服务器，并且只有它所需要的服务。

Tip 可通过浏览 www.cert.org 来了解你的操作系统的脆弱性。CERT（Computer Emergency Response Team）和 Internet 社团一起促进人们对安全问题的重视，并答复有关的问题。它也进行旨在提高计算机系统安全的研究。

12.2.3 Web 服务器软件

很明显，你正在使用 Apache 作为 Web 服务器软件，不然你也不会读本书。确保你所用的是最新版本的 Apache，最好自己编译 Apache 文件而不是使用二进制发行版。

查阅第 2 章怎样配置并编译 Apache 二进制文件。

Apache 是自由软件，因此从可靠的渠道获得该软件就显得很重要。不要随便从哪个 Web 站点或 FTP 站点下载 Apache 二进制文件或源码，一定要先检查正式站点 www.apache.org。将来的 Apache 源码将由 PGP（Pretty Good Privacy）签名。事实上，目前的 Apache 版本中有一个 PGP key-ring 文件，该文件中包含了大多数积极的 Apache 开发人员的 PGP key。可以使用该文件来认证 PGP 签名的 Apache 发行版。由 Apache 开发人员所发出的 Email 宣布或安全忠告也可能用 PGP 签名以保证可靠性。

Apache PGP 签名在发行版顶层的 KEYS 文件中。如果你信任获得 Apache 的站点，可以把这些 key 加到自己的 key-ring 中去。如果不知道怎样在你的系统上使用 PGP，可以在下面的 Web 站点学习它：

www.pgp.com

有些版本随其发行的还有含有 PGP 签名的文件。这些签名存在带 .asc 扩展名的文件中（如 apache_1.3.0.tar.gz 的签名存在 Apache_1.3.0.tar.gz.asc 文件中）。如果已经将 KEYS 文件内容加到 key ring 中，就可以使用这些签名来做有效性检查。

最新的 Apache 版本现在尚未发现隐错，然而它能够同外部模块一起工作的能力却为它带来了安全隐患。Apache 模块中有许多是由全球各地的人写的。确保所使用的模块位于正式的 Apache Web 站点，或者在 Apache 模块登记处登记过，该登记处位于：

www.covalent.net/module_registry/

千万不要在正式使用的站点中使用试验性模块。事实上最好的做法是：首先决定哪些 Apache 模块真的需要，然后配置并编译 Apache。这是一个好办法，这样就有了一个更小，更快，因此也更安全的服务器。

当配置 Apache 时，要极其注意随之而来的安全配置问题。基本的概念是：禁止使用没有用的东西。这样做，就采取了预防性安全措施，并减少了安全风险。

12.2.3.1　Apache 的用户和组

从前面的内容可以知道 Apache 既可以独立运行，也可以作为由 inetd 守护进程运行的服务。如果选择将 Apache 作为 inetd 服务，就不必操心用户和组命令。但是如果将 Apache 作为独立的服务器，那么就要保证为 Apache 生成了专用的用户和组。不要使用 nobody 用户或 nobody 组，尤其当你的系统已经定义了这两个用户和组。因为有可能其他的服务或系统其他地方用到它们，这将带来管理上的麻烦。相反应当使用前述命令来为 Apache 建立新的用户和组。

当使用 Apache 专用的用户和组时，Web 内容的权限管理就变得相当简单，所需做的只要保证 Apache 用户对 Web 内容有读权限。如果生成了一个内含能写数据的 CGI 程序的目录，只需为 Apache 用户设置写权限。

12.2.3.2　保护 ServerRoot 和日志目录

确保除 root 用户外任何人对 ServerRoot 目录均无写权限（特别是日志目录及其中的文件）。不要给 Apache 用户或组对日志目录的读或写权限，否则将带来重大的安全问题。

12.2.3.3　禁止缺省访问

严格的安全模型不允许有缺省访问。因此应养成习惯，一开始就禁止任何人访问。只允许特定用户对特定位置的访问。为了实现无缺省访问，应在 Apache 配置文件中使用下列配置段：

```
<Directory/>
order  deny, allow
Deny from all
</Directory>
```

这段配置首先禁止所有的访问。现在如果想允许对特定目录的访问，可使用 </directory>容器再次打开那个目录。例如，如果想允许访问目录/www/mysite/public/htdocs，添加下列配置语句：

```
<Directory /www/mysite/public/htdocs>
```

```
oreder deny, allow
Allow from all
</Dircctory>
```

只打开需要的目录。这是一种预防性措施，因此极力推荐采用此方法。

超星阅览器提醒您：
本文受版权保护
请尊重相关知识产权！

12.2.3.4 禁止用户越限

如果不想使用户利用在目录中的针对目录的配置文件.htaccess 来越过服务器配置，使用下列方法使该功能无效：

```
<Directory/>
AllowOverride None
Options None
allow from all
</Directory>
```

这样就不允许用户越过服务器配置，并且事实上提高了服务器速度。因为服务器无需对每个请求查找针对目录的访问控制文件，因此提高了运行速度。

12.2.4 内容

任何关注 Web 安全的人都需要评估保护 Web 内容的手段和方法。当决定怎样保护内容时，可以采用下列指导方针：

◆Web 内容文件，如 HTML、图像、声音和录像剪辑等对 Web 服务器是只读的，除了内容拥有者外任何人对该目录和文件不能有写访问权限。

◆Web 不显示的任何目录或文件都不应该放到 Web 内容目录内。这些目录或文件应删除或移到其他位置，因为即使没有从可视内容到这些文件的链接，它仍旧能被他人访问。

◆任何动态内容生成程序，如 CGI，所生成的临时文件，都应放到对该程序有写权限的子目录中。该目录必须位于 Web 内容区外以保证应用程序内的错误不会将已有的内容文件删掉。

◆为了进一步澄清内容的版权，网页的内容中应该插入相应的版权声明。插入的版权声明应尽可能放在文档的起始位置。例如，在 HTML 文件中可以用一对 Comment 标记将版权信息嵌入到文件起始位置。如果打算经常更新版权信息，可以考虑使用#include 命令的 SSI 解决方案。详情请参照第 7 章。

◆如果有大量需要保护版权的图像内容，则应该研究一下刚刚出现的水印技术，这种技术在图像内隐含地内建了一些信息以保护版权。这种想法是：当查到某些站点未经允许就使用了你的图形内容，你可以通过隐藏起来的信息来证明这是偷窃得来的。如果这些信息和你的水印 ID 相匹配，你可以明确地找出窃贼并采取适当的法律步骤。但是注意，目前可以使用的水印工具都很值得怀疑，因为许多程序能够轻易地删掉原有版权拥有者的水印。然而，如果你很关注你的图形内容的版权的话，这项技术还是值得研究一下的。

除了这些指导方针外还要想到 Web 蜘蛛和机器人。所有搜索引擎，比如 Yahoo！、AltaVista、Excite 和 Infoseek 都使用自动机器人程序搜索 Web 站点和索引内容。这通常是需要的，但偶尔地你可能会想停止这些机器人访问你的 Web 站点的某一部分。

如果你的 Web 站点的某部分内的内容更新的频率很高（如每天一次），你可能并不想搜索引擎为它生成索引项。为什么呢？ 因为当用户在搜索引擎站点上找到了一个旧内容的链接然后点击这个链接却发现这个链接不存在，客户就会很不高兴。那个客户就可能去到下一个链接而不再访问你的站点。

也可能有其他的原因使你不想让搜索引擎为你的内容（或一部分）生成索引。机器人程序可能会有意无意地影响它所穿越的网络或 Web 站点。偶然地，搜索引擎也会因为快速地请求了太多文档而“淹没”Web 站点。许多人正努力地为 Web 机器人制订 Internet 行为标准。目前的版本《机器人例外协议》允许 Web 站点的管理员将 robots.txt 文件放到 Web 站点上以指出机器人不许搜索的位置。例如，很大的位图图像档案对在 HTML 网页建索引的机器人来说应该是没有用处的。为机器人提供这个文件既浪费你的服务器，也浪费机器人的资源。

当前的协议还只是自愿的。机器人的开发人员在使用的过程中获得了大量经验，这也有助于机器人工作规则的演进。大多数流行的搜索引擎都遵守《机器人例外协议》。

当 Web 机器人访问名为 www.somesite.com 的站点时，它首先检查下列 URL 存在与否：

```
http://www.somesite.com/robots.txt
```

如果这个 URL 存在，机器人就会根据指导它怎样为该站点建立索引的命令将该文件的内容映射过去。作为 Web 服务器管理员，你可以为你的站点生成有意义的命令。注意：每个站点只有一个/robots.txt 文件。该文件内存有类似于下面例子的记录：

```
User-agent: *
Disallow:/cgi-bin/
Disallow:/tem/
Disallow:/~sam/
```

第一条命令告诉机器人接下来的命令任何机器人都要考虑。接下来的三个命令（Disallow）告诉机器人不要访问命令中提到的目录。需要注意的是对于每个想排除在外的 URL 前缀都要一个单独的 Disallow 命令，不能像下面这样：

```
Disallow:/cgi-bin/  /tmp/  /~sam/
```

并且也不能在记录中有空行，因为空行是用作多记录的限界符。User-agent 或 Disallow 命令中都不支持正则表达式，User-agent 域中的“*”是一个特殊值，表示任何机器人。特别不能使用类似下面的行：

```
Disallow:/tamp/*
```

或

```
Disallow:*.gif
```

任何没有明确不允许访问的目录或文件都被认为是机器人可访问的（下面举一些例子）。

为了从整个服务器中排除所有机器人，可用下列配置：

```
User-agent:*
Disallow:/
```

为了允许所有的机器人全面进行访问，可用下列配置：

```
User-agent:*
Disallow:
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

删除 robots.txt 可以达到同样的效果。为了排除某个名为 WebCrawler 的机器人：

```
User-agent:WebCrawler
Disallow:/
```

为了只允许某个名为 WebCrawler 的机器人访问站点，配置如下：

```
User-agent:WebCrawler
Disallow:
User-agent:*
Disallow:/
```

为了不让文件/daily/changes_to_often.html 被机器人建索引，配置如下：

```
User-agent:*
Disallow:/daily/changes_to_often.html
```

目前还没有 allow 命令。

12.3 CGI 风险及解决办法

Web 最大的安全风险来自 CGI 应用程序。CGI 本身并不是不安全的。但是，写得很糟的 CGI 应用程序是 Web 安全漏洞的主要来源。CGI 规范的简单性使很多经验不多的程序员能够写出 CGI 应用程序。这些缺乏经验的程序员并没有意识到互联网的不安全方面。他们开发的应用程序能够工作，但是也给运行应用程序的系统带来了隐藏的安全漏洞。

下面是 CGI 程序带来的三个最可能的安全风险：

♦信息泄漏——这种风险会帮助黑客侵入系统。黑客对一个系统所掌握的信息越多，它就会越有可能成功侵入系统。

♦通过 CGI 程序执行系统命令——很多情况下远程用户成功地欺骗了基于 HTML 表单的邮递程序脚本来运行系统命令或让系统给出保密的系统信息。

♦消耗系统资源——一个写得很糟的 CGI 应用程序可能会大量的消耗系统资源而使服务器基本上无法对其他请求作出反应。

当 Apache 服务器将 CGI 应用程序作为无特权用户运行时，当然应该仔细开发和安装这些程序。但是，即使这样它们也可能带来安全风险。接下来谈一下潜在的风险以及怎样预防。

12.3.1 预防用户输入带来的风险

CGI 应用程序带来的大多数安全漏洞来自用户输入。下面几节中将讨论用户输入怎样带来的风险和怎样预防。

12.3.1.1 用户输入使 CGI 应用程序崩溃

最大的危险来自用户输入。一种类型的用户输入问题发生在：程序在读输入时允许缓冲区溢出。这种情况可能导致主要的安全漏洞。因为缓冲区溢出时程序会崩溃，很多熟练

的黑客就能够利用崩溃程序并获得对系统的访问。在像 C 和 C++这类语言中字符缓冲溢出实际上常出现。

下面是 C 程序的示例。假设用户输入最多为 1024 个字符（即 1024 字节长），因此定义一个 1024 字节的数组就足够了。当用户输入了大量数据时，这个错误假设就会引起程序崩溃或不能正常工作。

超星提醒您：使用本复制品请尊重相关知识产权！

```
# include <stdio.h>
# include <stdlib.h>
static char query_string[1024];

char* POST(){
#
# Function to read input that has been POSTED from an HTML
# form.

# Local integer variable
int size;

# use getenv() function to get the value of the environment
# variable
# CONTENT_LENGTH which holds the byte count for the form input
# data stored in
# the STDIN. Convert the resulting string value of genet() to
# an integer using
# the atoi() function and return the value to the size
# variable.
size=atoi(getenv("CONTENT_LEIGTH"));

# Read the input data from the standard input.
fread(query_string,size,1,stdin);

# Return the string buffer holding the data.
return query_string;
}
```

当用户键入的数据大于 1024 字节时，就会使程序崩溃，并增加安全问题发生的可能性。有些情况下，黑客利用程序崩溃来远程执行命令。

解决这个问题的办法是：确保使用 malloc( )或 calloc( )函数来动态分配存储数据的内存（query_string）。

12.3.1.2　用户输入使系统调用不安全

当用户请求的系统调用打开 subshell 来处理命令时，就会带来另一个用户输入风险问题。例如，在 Perl（一种广泛使用的 CGI 编程语言）中可以使用 systm( )、exec( )、piped open( )和 eval( )函数来实现这种调用。同样，C 语言中的 popen( )和 system( )函数也是潜在的安全威胁。所有这些函数/命令都调用 subshell（如/bin/sh）来处理用户命令。

注意，即使 shell 脚本使用 system( )、exec( )调用也可能会为黑客打开方便之门。还

要注意的是在 shell 解释程序中和 Perl 中用来获取程序输出正文字符串的单引号也很危险。

为了说明小心使用系统调用的重要性，来看一下下面这段看似安全的 Perl 代码。

```
#!/usr/local/bin/perl
#
# Purpose:to destramont security risks in poorly written CGI
scripts
#

# Get the domain name from query string environment variable
#
my $domain = $ENV{'QUERY_STRING'};

# Print the appropriate content type. Since whois output is in
# plain text
# we choose to use text/plain as the content-type here.
#
print "Content-type:text/palin\n\n");

# Here is the bad system call
system("/usr/bin/whois $doamin");

# Here is another bad system call using back-ticks.
#
# my $output = '/usr/bin/whois $domain';
#
# print $output;
exit 0;
```

这段 Perl 脚本可以认为是基于 Web 的 WHOIS 网关。如果该段脚本称为 whois.pl，并且将它存放在名为 www.notsecured.com 的站点的 cgi-bin 目录下。用户可以像下面这样调用该脚本：

```
http://domain/cgi-bin/script.pl?domain anydomain.com
```

这一脚本通过 QUERY_STRING 变量使用 anydomain.com 作为$domain 变量的值。然后将$domian 的值作为参数启动 /usr/bin/whois 程序。这个请求会从 InterNIC 维护的 WHOIS 数据库中返回数据。这个想法很好，但是这个脚本却是即将发生的灾难。考虑运行下面的 HTTP 命令：

```
http://domain/cgi-bin/script.pl?domain nitec.com;ps
```

该请求会在名为 nitec.com 的域上进行 WHOIS 查找，并同时提供用来显示进程状态的 Unix ps 实用程序的输出。这就暴露了请求方不应看见的系统信息。使用这一技术的任何人都能够找到大量关于你的系统的信息。假如用 df（打印磁盘空间利用的综合信息的 Unix 实用程序）来代替 ps 命令，那么任何人都能够知道你的磁盘有什么分区以及使用情况。现在你能想象这个安全漏洞可能带来的危险。

这给我们什么教训呢？教训是不要信任任何输入信息，不要滥用系统调用。下面介绍

一下怎样达到这两个目的。

确保用户输入安全的方法有两个。一是扫描输入以检查输入是否有不合法字符然后替换掉。例如对上面的 whois.pl 脚本，可以加入下列行：

```
$domain=~ s/[\/;\[\]\<\>&\t]//g;
```

这将会删去非法字符。这种方法虽然通用但不明智，因为它需要程序员意识到会带来麻烦的字符的所有可能组合。如果用户输入了程序员没有预先想到的字符，程序可能会以程序员不希望的方式被利用。

更好的方法是定义一系列可接受字符，并且替换或删除任何不可接受的字符。有效输入值列表在典型情况下应当是可预知的、准确定义的可管理的集合。

这种方法并不要求程序员捕获所有不可接受字符，因此也就没有出错的余地。在这种方法中，程序员必须且只须确保可接受的字符是可标识的，这样他就不必关心攻击者企图越过安全检查所试验的字符是什么。

基于这一想法，前面的 Perl 脚本可以重写，只允许预定义的字符通过，如：

```
#!/usr/local/bin/perl
#
# This is a better version of the previous whois.pl script
#
# Assign a variable the acceptable character set for domain
# names
#
my $DOMAIN_CHAR_SET='~a-zA-Z0-9_.';

# Get the domain name from query string environment variable
#
my $domain = $ENV{'QUERY_STRING'};

# Now remove any character that does not belong to the
# acceptable character set.
$domain =~ s/[$DOMAIN_CHAR_SET]//g;

# Print the appropriate content type. Since whois output is in
# plain text we choose to use text/plain as the content-type
# here.
#
print "Content-type:  text/plain\n\n";

# Here is the system call
system("/usr/bin/whois $domain");

# Here is another system call using back-ticks.
#
# my $output = '/usr/bin/whois $domain';
#
```

```
# print $output;
exit 0;
```

$DOMAIN_CHAR_SET 变量包含可接受字符集。程序搜索用户输入变量$domain 看其中是否含有不在集合中的字符串，不可接受的字符串将被删除。

处理用户输入最好的方式是为用户输入建立规则（即期望收到什么以及如何确定所收到的是可接受的）。如果希望的输入是一个 email 地址（并不需要盲目地扫描搜索 shell 的元字符），可用下面的正则表达式来检查输入作为可能 email 地址的有效性：

```
$email=$input {'email-addr'};
if ($email=~ /^[\w-\.]+\@[\w-\.]+$) {
        print "Possibly valid address."
}
 else {
    print "Invalid email addres.";
    }
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

仅检查用户输入并不够，还要注意如何调用外部程序（例如，在 Perl 中有许多方法可以调用外部程序）。这些方法包括：

用单引号可以获取外部程序的输出，如：

```
$List = '/bin/ls -l /etc';
```

该行命令获取 /etc 目录的列表。也可以为一个程序打开管道，如：

```
open (FP, "| /usr/bin/sort");
```

还可以用 System( )调用外部程序并等待返回：

```
System "/usr/bin/lpr data.dat";
```

用 exec( )可调用外部程序并且无需返回：

```
exec "/usr/bin/sort < data.dat";
```

如果这些语句涉及可能包含 shell 元字符的用户输入，它们都有风险。对于 system( ) 和 exec( )，它们存在模糊的语法特征，可不通过 shell 就直接调用外部程序。如果将参数传给外部程序（不是长的字符串，而是列表中分开的元素），Perl 就不会通过 shell 并且 shell 的元字符也不会带来不希望的副作用，如：

```
System "/usr/bin/sort", "data.dat";
```

使用这项功能无需通过 shell 就能打开一个管道。通过调用 open 字符串序列“-|”，就能为 Perl 建立一个拷贝并为该拷贝打开一个管道。然后子拷贝立即使用 exec 函数调用的第一个参数生成另一个程序。

为了不打开 shell 就能从管道中读数据，可以用字符序列“-|”作类似的事：

```
open (GREP, "-|") || exec "/usr/bin/grep", $userpattern,
$filename;
while (<GREP>) {
   print "match:$_";
   }
close GREP;
```

这种形式的 open( )比管道 open( )更安全，因此应该尽量使用这种形式。

注意，在 Perl 中有许多其他能够调用外部程序并向其谎报名称的模糊功能。有些程序根据调用时使用的名称不同其执行的功能差别很大，前述的功能对于调用这类程序很有用。语法如下：

```
system $real_name "fake_name", "argument1", "argument2"
```

黑客玩的一个花招是改变 PATH 环境变量的值使它指向他想让你的脚本执行的程序——而不是你希望的程序。你应该用全路径名调用程序，而不应依赖于 PATH 环境变量。也就是不用类似下面的 Perl 代码：

```
system ("cat /tmp/shopping.cart.txt");
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

而用：

```
system "/bin/cat", "/tmp/shopping.cart.txt";
```

如果执意要用 PATH 变量，那么要在 CGI 应用程序的开始设置，如下：

```
$ENV{'PATH'}="bin:/usr/bin:/usr/local/bin";
```

当调用外部程序时即使不依赖于路径，但是调用的程序也可能会用到它。因此，应该无论何时都应该把上面一行代码放到脚本的开始，只要使用不合法性检查。注意，必须将上一行改成想搜索的目录列表。另外还要注意，一般情况下将当前目录（.）放入 PATH 中并不是好主意。

Perl 能够处理不合法输入：

CGI 应用程序中最常见的问题是无意中将未检查的用户变量传给 shell。Perl 提供了不合法性检查机制来防止这种情况发生。任何从程序外部对变量赋值的数据（包括来自环境变量、标准输入和命令行的数据）都被认为不合法，并且不能够在程序外部用来影响任何变量。不合法数据可能会散布开，如果用不合法的变量设另一个变量的值，第二个变量的值就也不合法了。不合法变量不能用在 eval( )、exec( )、system( )或管道类型 open( )调用中。如果企图这样做的话，Perl 就会给出警告信息并退出。如果没有明确地设置 PATH 环境变量，当企图调用外部程序时程序也会退出。可以在 Perl 5 解释程序的调用行中使用“-T”标志打开不合法性检查：

```
#!/usr/local/bin/perl -T
```

像前面所提及的那样，一旦一个变量值不合法，Perl 就不会在 system( )、exec()、管道类型 open()、eval()或单引号命令（或任何从程序外部影响程序的其他函数，如 unlink）中使用该变量。即使扫描了其中的 shell 元字符串，并使用 tr///或 s///命令删掉了元字符，但还是不能用。使不合法变量变为合法的唯一方法是在其上执行模式匹配操作并抽取匹配的子串。

例如希望一个变量包含 email 地址，可以用下面的方式抽取它的合法的拷贝：

```
$e-mail=~/([\w-\.]+\@[\w-\.]+)/;
$untainted_address=$1;
```

因此打开 Perl 的不合法性检查是个好主意。

最根本的原则是避免直接将用户输入传给外部程序，因为里面可能有能给系统造成损害的隐藏命令。尽管避免使用能够打开子 shell 的系统调用，每次都扫描输入中的 shell 命令元字符，然后将它们从输入中删除。

12.3.1.3 用户看隐藏的数据

必须认识到与输入有关的另一种形式的危险，需考虑隐藏类型的 HTML 输入字段的问题。需要大量用户输入的 CGI 应用程序通常向用户请求一些输入，然后基于这些输入作一些处理，之后再向用户请求输入。因为 CGI 程序是对每次请求作出响应，因此它并不能维护用户上一次交互中所输入的数据。为了解决这个问题，许多 CGI 程序员使用 HIDDEN 输入字段将前面输入的数据存在输出 HTML 中，HTML 隐藏输入记录形式如下：

```
<INPUT TYPE=HIDDEN NAME="variable" VALUE="value">
```

使用这种方法没有什么错，但是要记住：虽然它们假设被认为是隐藏的，实际上并不是。任何用户都可以使用浏览器中的 View Source（查看源文件）选项看到这些数据。一定要保证在 CGI 应用程序调用之间保存的数据不是敏感数据。换言之，不要在这些字段中保存路径、文件名、用户名或口令等数据。

仔细检查了用户输入，并保证以对系统威胁最小的方式使用系统调用，则可能会想这是不是所要作的一切呢？ 不，正像前面所说的：保证安全永无止境。它是一个长期的学习的过程，正当学习克服一个问题时，另一个问题就又出现了。下面几节中将讨论常见的滥用 CGI 应用程序的问题。

12.3.1.4 用户输入导致拒绝服务

最常见的安全攻击是拒绝服务（DOS）的攻击。这种攻击能使系统对合法用户或付费用户不能反应或不能使用。这种攻击也可以用在 CGI 应用程序上。

大多数 CGI 应用程序接收用户输入数据，执行一定的处理然后返回输出结果。因为通过 Web 任何人都能够运行这些程序，因此它们也是 DOS 攻击的对象。一个坏心眼的用户只需输入 CGI 应用程序所需的数据并调用你的 Web 服务器上的 CGI 程序，这个程序就启动了。然后他就可以不断地调用这个或那个 CGI 程序以使你的 Web 服务器生成许多子服务器进程和 CGI 进程。

这样做可能会消耗大量的服务器资源并使它对合法用户的反应变慢。不幸的是这种攻击很容易从大学的计算机实验室发起，因为那里有大量的计算机和足够的带宽。很小的 Perl 脚本就能向你的服务器发出请求并不断地重复同一请求。如果从多台机器上同时这样做，就可能会使 Web 服务器速度变得极慢。那么怎样解决这一问题呢？

不幸的是并不能为此作很多工作，但一旦发生这种事故，就可以禁止发起攻击的主机，可利用<Limit..>容器来拒绝这些主机的访问。

也可以用 RLimitCPU 和 RLimitMEM 命令来调节 Apache 的 CPU 和内存使用。

详见第 10 章中关于使用<Limit..>容器的部分以及第 4 章中关于 RlimitCPU 和 RLimitMEM 命令部分。

12.3.1.5 在站点上公布用户输入

在 Web 上，另一种常见的 CGI 破坏行为时有发生。在这些站点上允许任何用户写文本，并在以后到 Web 站点上公布。例如，使用基于 Web 的 CGI 讨论组或者 CGI 客户留言簿都可能成为这种破坏行为的牺牲品。

如果在你的 CGI 程序中，允许用户输入任何内容，并到你的 Web 站点上公布，千万保证在用户可能送入的任何输入中禁止 HTML 标记项。在 Perl 中，禁止使用 HTML 标记的最简单方式是将小于号（<）用“&lt”代替。用这种方法，当在浏览器中显示时，浏览器并不解释标记，而是显示实际的 HTML 标记。下面的 Perl 代码即可完成这一功能：

```
$user_input=~ s/</&lt;/g;
```

许多攻击者在发现可能写客户留言的站点时，通常都会键入能够链接到淫秽站点或大图像的 HTML 标记，使真正用户取得 Web 页时耗费大量的时间。也有可能在客户留言簿中键入一段 Java 脚本，这段脚本可能会给合法的用户带来极大的不安。将小于号换为“&lt”可以阻止所有这些恶作剧。

12.3.2　用 wrapper 减少 CGI 风险

减少 CGI 风险最好的办法是根本不用 CGI 程序。然而，在动态 Web 内容时代，这样做可能等于自杀！可以将所有 CGI 程序集中到某个位置，密切关注这些程序的开发，保证程序尽量不犯前面的错误。

在许多情况下，尤其是在 Internet 服务提供商的系统中，所有 Web 站点的用户都想获得 CGI 访问。在这种情况下，利用拥有 CGI 程序的用户的用户 ID 来运行 CGI 程序是个好办法。在缺省情况下，Apache 运行的 CGI 程序使用 Apache 的用户 ID。如果使用程序所有者的用户 ID 运行这些程序，那么就可能将损失限制在允许这一用户 ID 访问的范围内。换言之，一个很糟的 CGI 应用程序使用用户 ID，而不是使用 Apache 服务器的用户 ID 来运行，只能损害该用户的文件。这样做有两方面的好处：因为可能的损害只影响到他自己的内容，负责 CGI 程序的用户现在会更加小心。现只用一种措施就既增加了用户的责任心和风险意识，同时也限制了潜在损害影响的范围。为了使用不同于 Apache 服务器的用户 ID 运行 CGI 程序，需要一种称为 wrapper 的特殊 CGI 程序。wrapper 程序允许拥有该文件的用户而不是 Apache 服务器用户来运行 CGI 程序。一些 CGI wrapper 程序在运行所请求的 CGI 程序之前会做一些其他的安全性检查。

下面将讨论两种流行的 wrapper 程序。

12.3.2.1　suEXEC

Apache 带有名为 suEXEC 的支持软件。它为 Apache 提供了使用不同于 Apache 用户 ID 的用户 ID 运行 CGI 和 SSI 程序的能力。

suEXEC 只能在 Unix 和 Unix 类平台上工作。

suEXEC 是一个 wrapper 程序。当 HTTP 请求运行 CGI 或 SSI 程序，而且管理员指定使用用户 ID 而不是 Apache 服务器用户 ID 来运行它们时就调用这一程序。当收到这样的

请求后，Apacle 服务器就会给 suEXEC 程序提供程序名和用户及组 ID。suEXEC 用给定的用户及组 ID 运行该程序。

在运行 CGI 和 SSI 程序前，suEXEC 执行一系列的检查以保证请求是有效的。除此之外，整个检查过程还检查下列方面：

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

♦调用所使用的参数数量是否合适。

♦它由允许运行 wrapper 程序的有效用户执行。在典型情况下，只有 Apache 用户才可运行 wrapper 程序。

♦它提供了本 Web 站点内的 CGI 程序或 SSI 命令（exec 命令的目标）。被请求的应用程序必须在当前站点由 DocumentRoot 命令指定的目录内。该目录或应用程序本身除所有者外是不能写的。应用程序不能是 setuid 程序或 setgid 程序，因为这类程序在运行时会改变用户 ID 或组 ID。

♦要求它使用有效的用户 ID 和组 ID 来运行 CGI 程序或 SSI 命令（exec 命令目标）。注意 suEXEC 不允许使用 root 用户或 root 组来执行任何程序。用户 ID 和组 ID 号都要低于配置时指定的用户 ID 和组 ID 号。这就禁止使用系统帐号和组号。

一旦成功地完成了上述检查，suEXEC 通过 setuid 和 setgid 调用将用户和组 ID 变为目标用户和组 ID。对用户组访问列表也进行初始化，使它包含所有该用户所在的组。suEXEC 通过建立安全的执行路径（在配置中定义）来清除进程的环境。它也只传递列于安全环境变量列表（在配置时生成）中的那些变量。suEXEC 进程然后变成目标 CGI 程序或 SSI 命令并开始执行。

这看起来工作量挺大——实际上确实很大，但是它也提供了更大的安全系数。

1. 配置和安装 suEXEC

可以在 Apache 源码的 support 子目录中找到 suexec.c 和 suexec.h 文件。为了配置编译 suEXEC，应该按以下几步操作。

步骤 1：编辑 suexec.h 头文件

像下面这样编辑 suexec.h 文件：

```
#define HTTPD_USER "www"
```

将 www 改为在 Apache 配置文件中 User 命令使用的用户 ID，这是允许运行 suEXEC 程序的唯一用户。

下面的宏定义了允许作为目标用户的最低用户 ID：

```
#define UID_MIN 100
```

换言之，低于这个数的用户 ID 不能够通过 suEXEC 运行 CGI 程序和 SSI 命令。应该看一下你的/etc/passwd 文件，并确保选择的范围不包括通常低于 100 的系统帐号。

下面的宏定义了允许成为目标组的最低组 ID：

```
#define GID_MIN 100
```

换言之，低于这个数的组 ID 不允许通过 suEXEC 运行 CGI 程序和 SSI 命令。应该检查一下/etc/group 文件以保证所选的范围中没有通常低于 100 的系统帐号。

下面命令在用户 home 目录下定义了含有 suEXEC 可执行文件的子目录：

```
#define USERDIR_SUFFIX "public_html"
```

Note　如果 UserDir 命令所指向的目录不在/etc/passwd 文件所指定的用户 home 目录下，SuEXEC 就不能运行。

超星阅览器提醒您：

如果有许多虚拟主机并且每个都有不同的 UserDir，那么必须将它们限定在一个父目录下，然后在这里为该目录指定目录名：

```
#define LOG_EXEC "usr/local/apache/logs/cgi.log"
```

这告诉 suEXEC 将每个调用和错误消息的日志信息写到哪里。

下面的宏为 Apache 定义了 DocumentRoot：

```
#define DOC_ROOT "/usr/local/apache/htdocs"
```

所有 suEXEC 可执行的 CGI 应用程序和 SSI 命令必须位于该目录树下，除非它是 UserDir 命令指定的目录的一部分。

下面的宏定义了 PATH 环境变量，该环境变量由 suEXEC 使用：

```
#define SAFE_PATH "/usr/local/bin:/usr/bin:/bin"
```

步骤 2：编译并安装 suEXEC 和 Apache

一旦编辑好 suexec.h 文件，就可以像下面这样编译 suexec 程序：

```
gcc  suexec.c -o suexec
```

该命令生成 suexec 可执行文件。如果想将该文件放到/usr/local/apache/sbin 目录之外的目录中，还必须得编译 Apache。在这种情况下，可编辑 httpdoc 文件并改变宏使其指向新 suexec 的位置，如下：

```
#define SUEXEC_BIN  "/usr/local/apache/sbin/suexec"
```

现在编译 Apache 并将新 Apache 可执行文件和 suexec 可执行文件放到适当的目录中。

步骤 3：设置 suEXEC 文件权限

因为只有 root 用户执行的程序才能够改变它们的用户 ID，因此必须像下面这样将 suexec 可执行文件的所有权设给 root：

```
chown  root  /path/to/suexec
```

还必须将 setuserid 位设置如下：

```
chmod  4711  /path/to/suexec
```

这样就完成了 suEXEC 的安装。现在讨论怎样使用它。

2. 允许使用和禁止使用 suEXEC

将 suexec 和新 Apache 可执行文件装在适当的位置后重启 Apache。不要用 HUP(-l)信号重启，原进程用 TERM 信号(-9)杀掉，然后再次启动 Apache。如果 Apache 配置正确，则会在控制台上看到下面的消息：

```
Configuring Apache for use with suexec wrapper.
```

如果没看到这条消息，检查 wrapper 是否安装在 httpd.h 文件中 SUEXEC_BIN 指定的位置。

如果看到了这条消息，那就说明正确安装了 suEXEC。现在如果想试一下 suEXEC，

只需在用户的 cgi-bin 目录中放一个 CGI 应用程序，然后通过下列 URL 访问该 CGI 程序：

```
http://www.yourdomain.com/~user/cgi-bin/scriptname
```

Apache 对所有 CGI 程序和 SSI 命令使用 suEXEC。这些程序或命令中都有波浪字符（~）。

下面的例子有助于理解 suEXEC 是怎样工作的。假设 Apache 服务器名为 wormhole.nitec.com，UserDir 设成 public_html，并且用户主目录是/home/kabir，public_html 目录位于该目录下。假设 test.cgi 文件位于~kabir/public_html 目录下。

```
#!/usr/local/bin/perl
print "Content-type:text/html\n\n";
print "Test of suEXEC <BR>";
print "PATH=$ENV{PATH}";
exit 0;
```

现在，当进入如下 URL 时：

```
http://wormholde.nitec.com/~kabir/test.cgi
```

它正如预期的那样运行。检查 cgi.log 文件（由 suexec.h 文件中的宏 LOG_EXE 指定），其中有下列信息：

```
[00:50:15 12-01-98]:uid:(kabir/kabir) gid:(kabir/kabir)
test.cgi
```

很好。现在看一下用 ls –l 显示“~kabir/public_html/test.cgi”命令会看到什么：

```
-rwxr-x-r-x 1 kabir kabir 85 Jan 12 00:50 test.cgi
```

现在看一看用下面的命令将脚本的所有权设给 root 用户会发生什么：

```
chown root test.cgi
```

如果此时企图运行脚本就会得到服务器错误信息，日志文件中有下列信息：

```
[01:01:30 12-01-98]:target uid/gid(500/500) mismatch with
directory (500/500)  or program (0/500)
```

此处程序所有者是用户 ID 为 0 的用户，并且组仍旧是 kabir（500）。因此 suEXEC 拒绝运行该程序。像所看见的一样，suEXEC 所作所为均在预料之内。为了保证 suEXEC 将会在其他的目录中运行 test.cgi 程序，在~kabir/public_html 目录下生成一个新的 cgi-bin 目录，并将 test.cgi 文件放到该目录下。将该目录和文件的用户和组所有权设给用户 ID kabir 和组 ID kabir 后，使用下面的 URR 访问脚本：

```
http: //wormhole.netic.com/~kabir/cgi-bin/test.cgi
```

运行结果和期望相同。注意这个脚本允许作 PATH 变量的值检查，因为在 suexec.h 文件中已经用 SAFE_PATH 宏设了安全路径的值，因此能够确保 PATH 变量的值恰好是所希望的。实际上确实如此。应该作类似的检查以保证 suEXEC 的设置正常。

如果有虚拟主机并且想用 suEXEC 运行 CGI 程序和/或 SSI 命令，则必须在<VirtualHost...>容器内使用 User 和 Group 命令。用这两个命令设置用户和组 ID，这些用户和组 ID 不同于 Apache 服务器正使用的 ID。如果在<VirtualHost>容器内只用了一个命令或一个也没用，服务器假设该虚拟主机使用服务器的用户 ID 和组 ID。

由于安全和效率的原因，对于虚拟主机请求 suexec 必须限于顶层文档根目录中，而

对于用户目录请求应位于个人文档根目录的顶层。例如配置了四个虚拟主机，应该将所有虚拟主机文档根目录放到 Apache 文档树结构中以利用 suEXEC。

现在看一下另一个名为 CGIWrap 的流行 CGI wrapper 程序。

12.3.2.2　CGIWrap

CGIWrap 很类似 suEXEC 程序，因为它允许用户在不损害 Web 服务器安全的前提下使用 CGI 应用程序。CGI 程序以文件所有者的权限运行。此外，CGIWrap 对 CGI 应用程序执行几种安全检查，并且如果有一种检查没通过就不执行应用程序。

CGIWrap 是 Nathan Neulinger 开发的，最新版本可以在站点 ftp://ftp.cc.umr.edu 找到。本书附带的 CD-ROM 中也有 CGIWrap 的最新版本。

CGIWrap 通过 HTML 文档中的 URL 调用。发布时 CGIWrap 配置成运行位于~/public_html/cig-bin/目录下的用户脚本。

1.配置和安装

CGIWrap 是以 gzip 压缩的 tar 文件发布，用 gzip 解压并用 tar 实用程序提取。

运行配置脚本，它就会提示回答一些问题。这些问题中大多数是很容易看懂的，但是也有一些问题不同于 suEXEC。它允许建立 allow 或 deny 文件，用来限制对 CGI 程序的访问。这两个文件有相同的格式，如下所示：

```
User ID
mailto:Username@subnet1/mask1,subnet2/mask2...
```

可以有单个用户名（并非数字的用户 ID）或者用户 mailto:ID@子网/掩码行，其中可以定义多个子网/掩码对，如：

```
mailto:Myuser@1.2.3.4/255.255.255.255
```

如果该行位于 allow 文件中（文件名由你指定），那么用户 kabir 的 CGI 程序可以被属于 206.171.50.0 网络上并且掩码是 255.255.255.0 的主机使用。这很有些类似于第 10 章中讨论的基于主机的访问控制。

运行配置脚本后运行 make 工具生成 CGIWrap 可执行文件。

2. 使用 CGIwrap

使用 CGIWrap 前，先将 CGIWrap 可执行文件拷到用户的 cgi-bin 目录下。注意该目录必须和配置过程中指定的目录一致。最简单的方法是让 CGI 应用程序使用~username/public-html/cgi-bin 类型的目录结构。

拷完 CGIWrap 可执行文件后，改变它的所有权和权限位如下：

```
chown root CGIWrap
chown 4755 CGIWrap
```

在 cgi-bin 目录下建立到 CGIWrap 的硬链接或符号链接 npb-cgiwrap、nph-cgiwrapd、cgiwrapd，如下：

```
ln [-s] CGIWrap cgiwrapd
ln [-s] CGIWrap nph-cgiwrap
ln [-s] CGIWrap nph-cgiwrapd
```

注意，在我的 Apache 服务器上，一般只指定.cgi 作为 CGI 应用程序的扩展名。因此

12.5　基本 HTTP 认证的风险

另一个内在安全风险来自 HTTP 协议和基本认证。在这种类型认证中使用的口令以 UU 编码格式通过网络。这种编码不是加密的，当口令通过网络时它并不能提供安全保护。这就是产生风险的原因。

一个网络黑客要对 UU 编码口令解码，首先必须有侦听设备从网络上所有的 IP 数据包中截取发送口令的那个 IP 数据包。虽然这并不是一般的黑客能够做到的，但它也并不是不可能做到的。

超星提醒您：使用本复制品请尊重相关知识产权！

因此，目前的 HTTP 认证方式应该认为是不安全的。对于关键性应用不推荐使用这种认证方式。如果需要更强有力的认证方式，那么可能不得不开发新的解决方案。可能得研究安全 Socket 层（SSL）来达到这一目的。SSI 在第 15 章中讨论。

12.6　安全策略考虑

管理员的安全策略对于达到健全的网络安全性是至关重要的。如果还没有安全策略，考虑将下列方面加到你的管理性安全策略中，作为网络安全管理的开端。

12.6.1　记录每一件事

日志文件记录了服务器对每次请求作出响应的有关信息。分析日志既能提供商业信息（如哪个 Web 页最受欢迎），又能提供安全信息。确保将 Apache 设置为将访问信息和安全信息均写入日志。在日志文件的帮助下可以跟踪用户访问了什么。养成随时查看日志的习惯——如果发现有什么异常更要仔细检查。错误信息日志可能是最重要的日志，更要密切注意。

有关日志的详细信息请看第 11 章。

12.6.2　维护 Web 站点的可信的拷贝

将你的 web 站点的可以信赖的拷贝放到更安全的主机上。如果你的 Web 服务器的公用信息的完整性受到了损害，则需要从别的机器复制可信的拷贝来恢复它。通常将可信的拷贝放在 Web 管理员能够访问的主机上（也许，组织内部负责建立和维护 Web 内容的人也可访问它）。它可能位于你的组织的内部网上。

为确保安全，可以使用强大的密码检查和（cryptographic-checksum）技术为每一个文件生成一个检查和。将文件可信的拷贝和检查和保存在具有写保护或可读之处。可以使用 MD5 加密方式为文件生成加密检查和。

12.6.3 从 Web 主机控制台管理站点

应该从 Web 主机控制台上管理站点，这样做能够减少 Web 服务器和管理员的工作站间的网络流量。但在许多情况下这样做并不可行（例如，在 Web 服务器不易被管理员访问的组织内）。当必须进行远程管理时，一定要保证使用健全的认证模式登录到服务器。如果使用基于 Web 的管理工具，要保证它并不使用基本 HTTP 认证。换言之，需要保证你的口令不是以非加密格式从工作站传输到 Web 服务器。并且还要配置 Web 服务器，使它只能从内部网接受单个主机的连接。

12.6.4 密切关注公用域 CGI 应用程序

当使用公用域 CGI 时，一定要保证组织内部有人（或请外面的顾问）完全理解这些代码。任何时候都不要使用来源不明的应用程序。到 USENET 上去找一找，看将要在 Web 服务器上安装的程序是不是有人发现其中的问题。如果可能的话，将它安装在试验性服务器上测试一下。测试阶段坚持监视日志文件。如果该应用程序使用邮件服务，也要监视邮件日志。

12.6.5 比较内容

入侵者通常替换、修改和删除系统上他们曾经访问过的的文件。为了再次非法访问系统，他们通常修改系统程序，使它们看似正常但却为入侵者留了后门。他们也修改系统日志文件以删除活动的痕迹：入侵者甚至可能在系统上新建文件。

因此，最好与可信的拷贝比较一下文件和目录的属性和内容。如果为文件生成了密码检查和，那么可以比较一下当前的拷贝和可信的拷贝的检查和是否有什么不同。

用 MD5 检查文件内容的完整性：

MD5 程序能根据文件内容生成唯一的 128 位的加密值，该值可以认为是一个用来检验文件内容完整性的极可靠的指纹。如果文件中有一位改了，该文件的 MD5 检查和也会改变。伪造一个与原来的文件具有完全相同的 MD5 检查和的文件被认为是极端困难的。

一组用于关键的系统、应用和数据文件的 MD5 检查和提供了以压缩方式存储信息的手段，用户可以使用这些信息周期性地检查这些关键文件的完整性。

有关 MD5 加密检查和程序的细节由 RFC 1321 提供。源码及附加信息可以通过 FTP 在下面的站点找到：

```
ftp://info.cert.org/pub/tools/md5
```

不要用 Unix 的 sum 程序生成检查和，因为它很不可靠。

如果发现了对系统不利的改变，则应认为系统已受了损害并采取适当措施。如果发现这种改变是必要的，则需重新建立可信拷贝以及有关文件的检查和。

12.7 有希望吗

超星阅览器提醒您：

关于安全和风险的这些讨论中，当考虑到没有明确的安全问题解决方案时，很容易感到沮丧。我个人认为：只要有了忧患意识和防范能力，就有可能赢得最后的胜利。

根据 John D. Howard 博士（《1989~1995 年 Internet 安全事故分析》一书的作者）的看法，除了拒绝服务攻击外，和 Internet 的规模比较，安全问题相对减少。他的研究是基于 CERT 的事故报告。该研究估计服务器发生安全问题但并不报告约占 0～4%，该研究同时指出：一般来说，一个 Internet 域内每年事故次数不超过一次。

该报告考虑了 7 年间一共 4567 次事故，报告的事故中有 5.9%是虚假警报。大多数事故（89.3%）是未授权访问，其中 24.1%用户帐号入侵、27.7%root 帐户入侵和 37.6%的访问企图。

和 Internet 主机数量增加相比，在研究期间每种类型的访问事故都在减少。相对于 Internet 主机增加，其中 root 级入侵减少 19%，帐号级入侵减少 11%，访问企图减少 17%。

12.8 小　　结

你已经有足够的技术来处理 Web 安全风险。本章中讨论了各种风险，并指出了一些解决办法。但是你也可能意识到：对 Web 安全问题的解决，还没有达到理想的境界——只要使用某种神奇的软件就能让我们（除坏家伙、黑客和破坏者外）高枕无忧。

Web 安全是一场和坏家伙（软件隐错、口令泄漏和其他讨厌的东西）之间的旷日持久的战争。因此只要时刻准备投入战斗，就不会输掉战争。还有许多组织，如 CERT，也站在你一边。防范安全风险的最佳方式是多做准备并不断增加风险意识。

下一章将讨论如何在 Apache 中使用流行的 CGI 脚本语言 Perl。

第四部分　实现高级特征

第 13 章　Apache 中的 Perl
第 14 章　Apache 中的代理服务器
第 15 章　Apache 中的安全 socket 层
第 16 章　重写 URL

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

第 13 章　Apache 中的 Perl

本章内容：

- 什么是 mod_perl
- 使用 mod_perl
- 为 mod_perl 写 Perl 模块
- 为 mod_perl 移植已有的 CGI 脚本
- mod_perl 与 SSI 的集成（mod_include）
- 用 Perl 设置 Apache
- 使用 mod_perl 必须注意的问题

近年来，Perl 已经成为最流行的 Web 客户程序开发语言。它主要用于开发 CGI 脚本。然而，由于它是一种解释语言，所以它也具有需要启动解释程序的不利因素。每次调用 Perl 脚本都需要装载 Perl 解释程序——perl。因此，启动 Perl 脚本要比启动编译成二进制可执行文件的应用程序慢。

第 9 章讲述了怎样利用 FastCGI 来加速基于 Perl 的 CGI 脚本。实际上，加速 CGI 脚本还有另一种方法，即使用 mod_perl——它是 Apache/Perl 集成项目的成果。mod_perl 不仅可以加速 CGI 脚本，还可以提供更多的功能。在本章中，主要讲述怎样和 Apache 服务器一起使用 mod_perl。

13.1　理解 mod_perl

Apache/Perl 集成项目的主要目标之一是把 Perl 程序语言的全部功能提供给 Apache 服务器。它导致了 mod_perl 模块的开发。将它和 Apache 和 Perl 编译、连接在一起，可以给服务器的 C 语言 API 提供面向对象的 Perl 接口。它允许 Perl 程序员用 perl 编写 Apache

模块。Apache-Perl 模块可以介入处理一个请求的各个阶段：如处理、头标解析、URI 转换、认证、授权、访问、类型检查、修补、日志记录和清除等。

正如前面所说，当使用 mod_perl 时，手中就有了 Apache API 的所有能力。现在来探讨怎样才能掌握这种能力。

当更新了 Perl，并在系统上安装了 mod_perl 后，你实际上为 Apache 服务器的 API 建立了面向对象的 Perl 接口。按照 Apache API，Apache 的请求对象保存服务器为请求提供服务所需要的所有信息。你的脚本怎样才能得到这样的重要对象呢？对象被作为参数传给 PerlHandler。例如：

```
<Location  /perl/>
SetHandler  perl-script
PerlHandler Apache::Registry
Options  ExecCGI
</Location >
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

在上面的配置中，PerlHandler 是 Apache::Registry 模块。这个模块必须实现称为 handler 的方法，它接收 Apache 的请求对象作为参数。这就允许这个模块去做使用 Apache 服务器 API 可以做的任何事情。

用 mod_perl 写的 Apache 模块能够完成用 C 写的 Apache 模块一样的功能。另外，嵌入服务器的常驻解释程序避免启动外部解释程序的开销，同时也避免了附加的 Perl 启动时间。例如，不需要像通常的 Web 服务器扩展那样启动一个独立的进程。扩展机制，例如通用网关接口（CGI），完全可以用 Perl 代码所取代。这些代码实现请求处理的产生响应的部分。为能更容易地运行已有的 perl CGI 脚本，mod_perl 包含了通用目标模块（Apache::Registry）。

现已了解了 mod_perl 可做的工作，下面将介绍 mod_perl 的可用模块以及怎样安装 mod_perl。

13.2　安装 mod_perl

世界各地的程序开发员正在不断开发各种 Apache/Perl 模块。在 http：//perl.apache.org/src/apache-modlist.html 这个网址下，可以看到最新的模块。以下是一些当前可供 mod_perl 使用的 Apache/Perl 模块：

- Apache::Registry——运行不变的 CGI 脚本
- Apache::Status——嵌入的解释程序的运行时状况
- Apache::Embperl——在 HTML 文档中嵌入的 Perl 代码
- Apache::SSI——在 Perl 中实现的 SSI
- Apache::DBI——维持持久的 DBI 连接
- Apache::DCELogin——获得 DEC 的登录环境
- Apache::AuthenDBI——通过 Perl DBI/DBD::*进行认证
- Apache::AuthzAge——基于年龄的授权

♦Apache::AccessLimitNum——使用请求次数限制用户的访问

♦Apache::Constants——在 httpd.h 中定义的常量

♦Apache::MsqlProxy——将 URI 转换为 mSOL 数据库查询

mod_perl 并不与标准的 Apache 发行版一起发布。你必须安装 mod_perl。但在安装前，需要保证认已满足以下的要求。

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

13.2.1 安装的要求

在你的系统上，应该有 Apache 源代码、Perl 5.003 或更高的版本、最新的 mod_perl 源代码。其中，可以从 www.perl.com 得到最新的 Perl 源代码发行版。如果还没有安装最新的 Perl，应该在做进一步工作之前安装它。

安装 Perl 只要解压缩源代码、运行配置脚本就可以了。配置脚本有大量的提示信息。如果对这些提示信息不确定，采用缺省值即可。使用：

```
make install
```

此命令编译最新的 Perl。接着用：

```
perl  -v
```

命令测试版本号。如果版本号与编译的版本相匹配，安装就成功了。

现在应该已经从 Apache Web 站点得到 Apache 源代码。从 http：//perl.apache.org 得到 mod_perl 源代码。

在光盘中也有最新的（到本书写作时为止）mod_perl 源代码发行版。但要保证使用的软件都是最新发行的版本。

如果已经得到了 mod_perl 源发行版，将它的源代码安装在你的 Apache 源发行版的同一层目录下。例如，如果把 Apache 源代码安装在 /usr/local/src/Apache 目录下，那么将 mod_perl 的源代码安装在 /usr/local/src 下面的 /usr/local/src/mod_perl_directory 目录中。

然后，可以使用如下命令：

```
zcat  mod_perl_1_10.tar.ga  |  tar  xf  -
```

将 /usr/local/src 目录下的 mod_perl_1_10.tar.gz 源代码解压缩，它将产生包含 mod_perl 的 /usr/local/src/mod_perl_1.10 目录。

13.2.2 编译和安装 mod_perl

在机器上有了 mod_perl 后，就可以准备编译和安装它了。如果要编译 mod_perl 源代码，需要把 mod_perl 的源目录作为当前目录，并运行命令：

```
perl  Makefile.pl
```

如果将 mod_perl 的源代码和 Apache 的源代码放在同一目录层下，当出现提示信息时采用缺省值即可。它将为 mod_perl 和 Apache 产生 Makefile。 下一步是运行命令：

```
make
```

make 命令执行后，将建立新的 Apache 二进制文件。可以通过运行如下命令来测试它：

```
make test
```

这将在端口 8529 启动 Apache 服务器，该服务器将运行在 perl Markfile.PL 进程的 UID 和 GID 下。如果由于某些原因这个端口已被使用，可以运行如下的带端口参数的 perl Markfile.PL 命令：

```
perl Makefile.PL  PORT=nnnn
```

其中 nnnn 是当前未被使用的端口号（大于 1024）。

如果测试成功，将看到“所有测试成功”之类的信息。接下来就可以安装 mod_perl 了。使用命令：

电子工业出版社提醒您：
使用本复制品
请尊重相关知识产权

```
make  install
```

安装 mod_perl 后，现在要做的就是用 Apache 源目录下的 httpd 的最新 mod_perl 版本替换原有的 httpd 可执行文件。可以从 Apache 源目录运行如下命令：

```
./httpd -v
```

这个命令将提供 Apache 服务器的版本号，同时提供 mod_perl 的版本号。如果没有，试一试下面的命令，看 mod_perl.c 是否列在输出中：

```
./httpd -l
```

如果 mod_perl.c 没有列在输出中，将不得不重复前面的安装过程。将当前已有的 httpd 重新命名为 httpd.old，再把新的 httpd 移到它原来的位置。这不失为一种好方法。这样，如果发生错误，可以恢复以前的版本。

13.3　通过 mod_perl 运行 Perl CGI 脚本

现在可以使用 mod_perl 了。首先讨论如何利用 mod_perl 的优点，提高现有的基于 Perl 的 CGI 脚本的运行速度。

从前面的章节中知道：为了运行 CGI 脚本，需要使用 ScriptAlias 命令定义 CGI 脚本的别名目录。然而，当使用 mod_perl 运行 CGI 脚本时，不能使用 ScriptAlias。需要使用 Alias 命令产生规则别名。例如，有一个称为 apache.nitec.com 的 Web 站点，它的 CGI 脚本放在 /www/apache/public/cgi-bin 目录下，则可以用下面的命令给它取别名 /perl/，如下：

```
Alias  /perl/  /www/apache/public/cgi-bin/
```

现在使用<Location...>容器，需要像下面那样进行设置：

```
<Location  /alias-to-cgi-script/>
SetHandler  perl-script
Perlhandler  Apache::Registry
Options  ExecCGI
</Location>
```

对于上面的例子，具体的设置如下：

```
<Location  /perl/>
SetHandler  perl-script
PerlHandler  Apache::Registry
```

```
Options  ExecCGI
</Location>
```

也可以使用 ScriptAlias 命令，如下：

```
ScriptAlias  /cgi-bin/  /www/apache/public/cgi-bin/
```

现在，可以通过传统的 CGI 方法使用下列 URL 访问名为 printenv.cgi 的应用程序：

```
http: //apache.nitec.com/cgi-bin/printenv.cgi
```

也可以通过 mod_perl 的 Apache::Registry 模块使用下列 URL 来访问它：

```
http: //apache.nitec.com/perl/printenv.cgi
```

怎样分辨这两种访问有什么不同呢？在 mod_perl 版本中标准 CGI 变量 GATEWAY_INTERFACE 值是“CGI-Perl/1.1”而不是“CGI/1.1”。同时，mod_perl 版本设置了一个新的变量 MOD_PERL。这个变量记录的是被使用的 mod_perl 的版本号。因此，如果想知道 Apache 是怎样运行你的 Perl 脚本的，可以使用如下的代码段：

```
If (exists  $ENV{'MOD_PERL'})  {
     Print "Running  as  a  mod_perl  application";
   }
else {
     print  "Running  as  a  CGI  script";
   }
```

在 mod_perl 方式下，Apache::Registry 模块常用于运行不变的 CGI 脚本。当请求下面的 URL 时，Apache::Registry 模块将读 printenv.cgi 脚本，并把它插入 Perl 子程序体中，最后执行它：

```
http://apache.nitec.com/perl/printenv.cgi
```

每个 Apache 子服务器进程都将编译这个子程序一次，并把它存在内存中，以供将来使用。如果在此期间 printenv.cgi 脚本有所改变，需要将这一脚本重新装入、重新编译、并再次存贮在内存中。

可以在服务器启动时预装 Perl 脚本。Mod_perl 提供给两个命令：PerlModule 和 PerlScript。可以使用这两个命令在启动时装载脚本。

为了在服务器启动时装载 Perl 模块，可以使用命令：

```
PerlModule  ModuleName...
```

例如：

```
PerlModule  Apache::SSI  Foo::Bar  Some::Other
```

这个命令将从 Apache 装载 SSI 模块、从 Foo 装载 Bar 模块、从 Some 装载 Other 模块。注意：使用 PerlModule 命令只能装载 10 个模块。如果需要更多的模块，可以使用 PerlScript 命令。

PerlScript 命令具有以下的语法：

```
PerlScript  /path/to/a/Perl/script
```

这个命令将在服务器启动时装载 Perl 脚本。如果在启动时需要装载许多模块，可以写一个这样的脚本：

```
use  CGI;
use  LWP::UserAgent ();
1;
```

这个脚本将装载 CGI 和 UserAgent 模块。如果想把模块保存在其他目录下，而不是在缺省的 Perl 模块目录下，需要给出模块的位置，命令如下：

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

```
use  lib  qw(/path/to/your/modules);
```

这里的/path/to/your/modules 应该用相应的模块目录的完整路径名代替。

为了使用 Perl 模块，必须使用前面提到的 PerlHandler 命令。例如，如果有 PerlModule 行如下：

```
PerlModule  Apache::Test
```

如果希望用这个模块为下面的 URL 服务：

```
http:://www.myserver.com/test/
```

那么可以定义一个别名：

```
Alias  /test/  /somewhere/some/directory/
```

接着，可以定义下面的</Location >容器：

```
<Location  /test>
SetHandler  perl-script
PerlHandler  Apache::Test
</Location>
```

这将告诉 mod_perl 你希望使用 Apache::Test 模块处理所有对特定的 URL 的请求。

Mod_perl 如何知道用什么方法（method）调入 Test.pm 模块呢？在默认的情况下，PerlHandler 命令期望得到的参数是子程序名。如果参数不是子程序名，它将参数当作模块名，并调用称为 handler 的特定方法。因此，对于上面的 PerlHandler 行来说，如果进来的是对特定 URL 的请求，mod_perl 将调用 Apache::Test::handler 方法。如果这个方法在模块中没有定义，将会产生一个错误。

只要在PerlHandler行中说明方法名，就总可以使用不同的方法名。例如，如果在Test.pm模块中有一个方法名称为 doit，并且想用它代替 handler，可以这样设置 PerlHandler 行：

```
PerlHandler  Apache::Test::doit
```

这样，mod_perl 就调用 doit 方法代替 handler。

前面已讲到怎样使用 Apache::Registry 模块将 CGI 脚本转变为基于 mod_perl 的脚本，对每一个请求 mod_perl 调用 Apache:Registry:handler 方法，然后由它运行请求的脚本。

13.4　移植 Perl CGI 脚本的问题

移植一个已有的基于 Perl 的 CGI 脚本将它转为基于 mod_perl 的脚本时，一定要注意下面的问题。

如果正在使用最新的 Perl 解释程序（5.004 版本或更高版本），那么你的大部分 CGI 脚本都不需要作任何修改就能够在 mod_perl 下运行。但是，如果使用的是 Perl 的旧版本，

那么在使用内置的读写函数时可能会遇到麻烦。此时，最好的方法是升级 Perl 版本。

在默认情况下，mod_perl 不发送任何 HTTP 头标。然而，使用下面的命令可以改变这种情况：

```
PerlSendHeader  On
```

现在，将发送响应行和 HTTP 头标。但你的脚本仍然需要发送如下的文本类型头标：

```
print  "Content-type:  text/html\n\n ";
```

如果使用的是 CGI.pm 或 CGI::Switch，并且用$query→header()方法打印头标，那么不必把 PerlSendHeader 设置为 On。

另一方面，在把非分析头标的 Perl 脚本（带有 nph-前缀的脚本）转换为基于 mod_perl 的脚本时，需要在每个输出文件处理的写或打印操作后强制进行输出清除。把$ | 变量设置为非 0 值就可做到这一点。

```
Local  $|  = 1;
```

注意，如果通常将 PerlSendHeader 设为 On，对于 nph-files 需要把 PerlSendHeader 设为 Off，如下所示：

```
<FilesMatch  */nph-*>
PerlSendHeader  Off
</FilesMatch>
```

13.5 为 mod_Perl 写 Perl 模块

现在，已经了解了在 mod_perl 中使用已有的 CGI 脚本应当注意的问题，下面就可以开始开发新的基于 mod_perl 的脚本了。为了让例子简单一些，下面讲述的是怎样创建一个很简单的 Perl 模块（一个面向对象的 Perl 脚本），这个模块实际上并没有做什么工作只是打印出环境变量。这个 Perl 模块叫做 Test.pm，如列表 13-1 所示。

列表 13-1：Test.pm

```
package Apache::Test;

# Tell Perl to show warning when a variable is used without
# a declaration.  This is done using 'use strict.'
use strict;

sub handler {
#
# Purpose: this method gets called by mod_perl.  So this is
# really the entry point into this module.  In other words,
# this is the traditional main function.
#

# Assign the request object (passed as a parameter) to a
# local variable.
```

```
   My $r = shift;

# Local temporary variable
   my $key;

# Assign a local associative array variable to the hash
# returned by the request object variable's cgi_env()
# methods
#
  my  %ENV = $r->cgi_env;

# Set the content-type of the output to text/html
#
   $r->content_type("text/html");

# Print out the HTTP headers
   $r->send_http_header;

# Now loop through the %ENV array and print out the
# key=value pair for each environment variable.
#
   foreach $key  (keys %ENV) {
      $r->print("$key = $ENV{$key} <BR>.");
     }

# Done
return 1;
}

# The following line is required for a Perl module.
1;
_END_
```

超星阅览器提醒
使用本复制品
请尊重相关知识产权!

在这个例子中，只有一个称为 handler 的子程序。如果像下面那样指定配置行：

```
<Location /test>
SetHandler perl-script
PerlHandler Apache::Test
</Location>
```

那么，无论什么时候当 Apache 接到包含/test/的请求时，都将调用 handler 方法。如果计划管理 Apache 请求过程，那么需要非常了解 Apache API。在若干请求阶段中，Apache API 允许模块进入并做一些事情。Apache 文档描述了请求的不同阶段和你的模块能做什么。在默认情况下，这些 hooks 在编译时是被禁止的。关于激活这些 hooks 的信息请参见 INSTALL 文档。

这里有一些 CGI 程序员开发 mod_perl 脚本应该注意的问题。首先看为什么 mod_perl

程序需要仔细进行程序设计，接着看一些写 mod_perl 程序的特殊编程问题。

许多 CGI 开发者不在整理代码上花费时间。这是因为 CGI 脚本运行时间很短，而且请求结束后它就消失了。但是 mod_perl 运行的脚本在内存中停留的时间很长。因此，开发这种脚本时需要仔细进行推敲。这里有几条原则供参考。

用严格的方式去写脚本。这要求写干净利落的语句。也就是说，在语句中不允许出现没有定义的变量。这种变量毫无用处，它既浪费内存又容易引起混乱。同时不要忘记使用 -w 开关。如果还不知道为什么要使用-w，建议去阅读 Perl 手册。你会注意到这一点是被再三强调的。下面一段话引自 Perl 的手册页：

无论何时你遇到难以理解的运行情况，试一试 -w 开关！！！

无论何时你没有遇到难以理解的运行情况，也试一试 -w！！！

虽然它不是必须的，但却被极度推崇。从长远观点来看，-w 开关可以节省更多的时间。

同时，使用用户输入数据时要非常小心。可以使用-T 开关启动 Perl 的不合法性检测，也可以在一个 Apache 设置文件中将 PerlTaintCheck 设置为 On。请阅读 mod_perl 发行版自带的 cgi_to_mod_perl 手册页。

还应该注意：在缺省情况下，不能使用 system()、exec()、open PIPE、| program 等系统调用。如果想使用它们，需要在 Perl 中有 sfio 支持。在默认情况下， exit()函数也不能使用； 但可用 Apache::exit()方法代替它。在 Apache::Registry 的脚本中，Apache::exit()将自动代替内置的 exit()。

除了以上这几点外，还有一些开发 mod_perl 脚本时需要考虑的原则。这些内容在下面的段落中讲述。

13.5.1 使用 CGI.pm 和 CGI::* 模块

如果在 CGI 脚本中使用 CGI.pm 模块，需要使用最新的版本。早于 2.36 的版本不支持 mod_perl。如果有最新的 Perl 版本和最新的 CGI.pm，就可以通过使用 use CGI 命令以平常的方式使用 CGI 脚本中的模块。但是，如果使用的是较老的 Perl 版本，就必须用 use CGI::Switch()命令，它将通过 Apache 类方法来处理输入/输出。

其他的 CGI 模块（CGI::*）只能在最新的 Perl 版本（5.004 或更高）中使用。

在使用 CGI::Request 模块和 SendHeaders()方法时，如果要结束一个请求，一定要调用$req_obj→cgi→done。

如果使用最新的 CGI::Switch 或 CGI.pm 和最新的 perl，你的脚本将只是从命令行中工作。

13.5.2 使用 Apache DB/DBM 认证模块

在使用 mod_ath_dbm 或 mod_auth_db 时，如果 Perl 的 Config.pm 中的 dbm 库的顺序不正确，就有可能出现问题。用下面的命令可以检查 dbm 库的顺序：

```
perl  -V:libs
```

在默认情况下，这些库与 Perl 连接，存储在 Config 模块中。如果 mod_perl 与 Apache 一起配置，ExtUtils::Embed 模块将返回这些与 httpd 连接的库，这样 Perl 的扩充功能可以在 mod_perl 下使用。然而，这些库在 Config.pm 中保存的顺序可能会把 Apache DB/DBM 模块弄乱。如果将 -lgdbm 或 -ldb 放在-lndbm 的前面，如下所示：

```
libs='-lgdbm  -ldb  -lndbm  -ldl  -lm  -lc'
```

则需要修改 Perl 的 Config.pm 模块，并把-lgdbm 和-ldb 移到最后。如果不知道修改哪个 Config.pm，使用如下的脚本来确定：

超星阅览器提醒您
使用本复制品
请尊重相关知识产权!

```
#! /usr/local/bin/perl
use  Config;
print  "Your Config.pm directory is"  $Config
{archlibexp}\n";
exit 0;
```

mod_perl 的接口足够灵活，在脚本装入后能够检测到脚本的变化。因此，如果修改了通过 mod_perl 装入的脚本，并不需要重启 Apache。但是，如果修改的是 Perl 库或通过 use 或 require 功能在你的脚本中使用的模块，那么 mod_perl 不能自动重新装入它们。在这种情况下，可以使用 Apache::StartINC 模块。

另一个要注意的地方是：如果在多个脚本中都包含了同一个 Perl 库（不是 Perl 模块），你需要 mod_perl 通过 Apache::Registry 功能运行这些脚本，此时只有一个脚本可以工作。这是因为当被请求的库被调入脚本时，它的子程序和变量将成为当前软件包的一部分。在典型的 CGI 环境中，它是主软件包。在使用 Apache::Registry 功能时，脚本被编译为唯一的软件包名称（基于 URL）。这意味者只有一个脚本可以使用被请求的库，mod_perl 不为其他的请求该库的脚本编译这个库。

解决这个问题的最简单的方法是重写 Perl 库使它成为面向对象的 Perl 模块。因为 Perl 模块都有一个软件包定义给出它自己的名称空间，所以被请求的模块（曾经是一个 Perl 库）可以被任何脚本使用。

阅读 perlmodlib 手册页、perlmod 手册页和相关的 Perl 文档，重写请求的文件，使它成为定义了方法接口的 Perl 模块。然而，如果不能把库重写为面向对象的 Perl 模块，请查阅 Perl 的输出函数和变量特征。

还要注意：如果想以始终与 Perl 静态连接的方式使用一个模块，该模块必须列在 Perl 的 Config.pm 的 static_ext 中，并且在建立 mod_perl 时与 httpd 连接。

mod_perl 也限制特殊名称（tokens）的使用。正如前面所说的，在 Apache::Registry 下运行的脚本不在默认的软件包（main）中运行。使用的名称空间基于被请求的 URL。因此，这些脚本不能包含特殊的名称。例如，不能用名称_END_表示程序的逻辑结束。此外，也不允许使用名称_DATA_。

13.6 Server Side Include 和 mod_perl 的集成

现已讨论了许多如何利用 mod_perl 的优点来开发 CGI 程序方面的内容。如果把 Perl 脚本作为 Server Side Include（SSI）脚本又如何呢？它也可以利用 mod_perl 的优点。

mod_perl 可以与 SSI 的模块 mod_include 集成。需要做的只是确保在建立 Apache 二进制文件时，在配置文件中有下面内容：

```
EXTRA_CFLAGS=-DUSE_PERL_SSI -I. 'perl -MextUtils:: Embed -ccopts'
```

或者确保在建立 mod_perl 时使用如下命令：

```
perl  Makefile.PL  PEAL_SSI=1
```

一旦建好了 Apache，就可以利用 mod_perl 的优点为你的 Perl SSI 脚本服务。如果想取代下面的 SSI 命令：

```
<!-#exec cmd="/some_directory_alias/your_perlscript" ->
```

可以在 SSI 页中使用如下内容：

```
<!-#perl sub="Apache::Include"
arg="/some_directory_alias/your_perlscript" ->
```

同时，需要在 httpd.conf 配置文件中用如下命令来预装脚本：

```
PerlScript  /path/to/your_perlscript
```

在前面例子中，SSI 调用中的 sub 键值可以是子程序名、模块名、某类方法名，或者是一个匿名的子程序调用（sub{}）。例如：

```
<!-#perl sub="Apache::Test" ->
```

在默认情况下，这条命令将调用 Test::handler 方法，只要 Test.pm 在服务器启动时通过 PerlModule 或其他方法预装。

如果想调用的不是默认的 handler 方法，需要同时给出方法名。例如，如果想调用 doit 方法而不是 handler 方法，就要这样改变前面的 SSI 调用：

```
<!-#perl sub="Apache::Test::doit" ->
```

如果需要给方法传递参数，可以像下面的例子那样把“argument1”、“argument2”作为参数传给方法。注意第一个参数总是请求对象的引用。

```
<!-#perl sub="Apache::Test::doit"  arg="argment1"  arg=
"argment2" ->
```

13.7 使用 Perl 配置 Apache

除了在 CGI 或 SSI 脚本中使用 mod_perl 所用到的优点外，mod_perl 还有不寻常的特征，热心的 Perl 程序员和 Apache 管理者一定会喜欢这个特征——它具有用 Perl 写 Apache 服务器配置的能力。

可以在 Apache 配置文件中加入<Perl>容器，如下所示：

```
<Perl>
```

```
# your Perl code goes here
1;
</Perl>
```

其中，最后一行（包含 1；）是需要的。你的代码可以是任何 Perl 脚本。<Perl>容器中的代码在特殊的软件包中编译，mod_perl 把配置信息传给 Apache 内核配置模块。在使用这个特征前，需要确保正确地建立了 Apache 和 mod_perl。为了给 Apache 服务器配置这个特征，需要这样配置 mod_perl：

```
perl  Makefile.PL  PERL_SECTIONS=1
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

mod_perl 和 Apache 被配置并重新编译后，就可在 Apache 配置文件中写 Perl 代码了。下面说明用于描述配置的语法。

取单个值的命令可以作为标量。例如：

```
User httpd
```

这个 User 命令的值是个字符串，因此可被写为：

```
<Perl>
$User = "httpd";
1;
</Perl>
```

下面是配置的例子：

```
<Perl>
$User = "httpd";
$Group = "httpd";
$ServerAdmin = 'kabir@nitec.com';
$MinSpareServers = 5;
$MaxSpareServers = 5;
$MaxClients = 40;
1;
</Perl>
```

需要多值的命令可以用列表表示。例如，PerlModule Apache::TestOne Apache::TestTwo 可以表示为：

```
@PerlModule = qw(Apache::TestOne  Apache::TestTwo);
```

容器可以改头换面来表示，例如：

```
<VirtualHost 206.171.50.50>
ServerName www.nitec.com
ServerAdmin kabir@nitec.com
</VirtualHost>
```

可以被表示为：

```
$VirtualHost{"206.171.60.60"}  = {
ServerName => 'www.nitec.com',
ServerAdmin => 'kabir@nitec.com'
 }
```

下面是另一个稍复杂的例子：

```
$Location {"/some_dir_alias/"} = {
```

```
  AuthUserFile => '/www/nitec/secret/htpasswd',
  AuthType => 'Basic',
  AuthName => 'Subscribers Only Access',
  DirectoryIndex => [qw(welcome.html  welcome.htm)] ,
  Limit => {
     METHODS => POST GET',
     Require => 'user reader'
   }
 };
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

可以用类似的方法定义其他的容器，例如<Directory>、<File>等。注意：为了使<Perl>容器能工作，需要在配置文件中定义 PerlScript 命令。如果不需要装入任何 Perl 脚本，可以像下面这样设置 PerlScript 命令来利用<Perl>容器：

```
Perlscript  /dev/null
```

建议只在出现下列一种或几种要求时才在配置文件中使用<Perl>容器：

♦你非常喜欢 Perl，不能想像既然在 Apache 配置中能用<Perl>为什么不用它。

♦运行了多个 Apache 服务器，并希望创建一组能用在所有 Apache 服务器系统上的配置文件（这正是希望使用的地方）。一个使用脚本的配置非常灵活。它可以读外部文件，使用循环来自动来配置等等。这样的配置只是受限于 Apache 管理者的编程能力。

如果为第一种情况这样做，那么应该在你的配置文件中用自己的风格写了有用的脚本。如果处于第二种情况下，并且不知道怎样利用 Perl 的这一特征，可以看一看下面的例子。

假设有三个 Web 服务器系统：host_a、host_b、host_c，其中 host_a 比 host_b 更加强有力，host_b 比 host_c 更强有力。现在，希望定义一个<Perl>容器，该容器允许为这三个主机创建单一的配置。下面就是这样一个 httpd.conf 文件：

```
<Perl>
# Get the host name using the UNIX hostname utility and store it
# in the $thisHost variable.
my $thisHost = '/bin/hostname';
if ($thisHost =~ /host_a/ )  {
# configuration for host_a goes here
$MinSpareServers = 10;
$MaxSpareServers = 20;
$StartServers    =30;
$MaxClients      = 256;
 }
elsif ($thisHost =~ /host_b/ )   {
    #  Configuration for host_b goes here
    $MinSpareServers =  5;
    $MaxSpareServers =  10;
    $StartServers    = 10;
    $MaxClients      =  50;
    }
```

```
else {
     # configuration for host_c goes here
     $MinSpareServers =  3;
     $MaxSpareServers =  5;
     $StartServers    =  5;
     $MaxClients      =  30;
     }
1;
</Perl>
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

为了使这个例子更有趣，假设这三个主机中的每一个都有不同的虚拟主机，我们希望这些虚拟主机以简练的方式配置。看看这个例子：

```
<Perl>
# Get the host name using the UXIT hostname utility and store it
in the $thisHost variable.

my $thisHost = '/bin/hostname';
my $thisDomain = 'mydomain.com';
my @vHosts = ();
my $anyHost;

if  ($thisHost =~ /(host_a)/)  {
   #  configuration  for host_a goes here
  @vHosts = qw(gaia, athena, romeo, juliet, shazam);
   }
elsif ($thisHost  =~  /host_b/ ) {
   #  configuration  for host_b goes here
  @vHosts = qw(catbart, ratbart, dilbert);
  }
else {
   # configuration for host_c goes here
   @vHosts = qw(lonelyhost);
   }

for $anyHost (@vHosts) {
  %{$VirtualHost{"$anyHost.$domainName"}} = {
      "ServerName" =>  "$anyHost.$domainName",
      "ServerAdmin" =>  "webmaster\@$anyHost.$domainName"
       }
  }
1;
</Perl>
```

Note　一旦为 Apache 服务器创建了合适的基于 Perl 的配置，就可以用下面的命令检查代码语法，确保代码的语法正确性：

```
perl  -cx  httpd.conf
```

13.8 使用 mod_perl 的特殊问题

前面讲述了 mod_perl 有什么优点和怎样利用它的优点。现在来看一看两个应该引起注意的问题。它们与系统资源和安全有关。

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

13.8.1 内存的使用

通过使用常驻内存的解释程序和在缓冲区中的脚本，mod_perl 可以使脚本执行得更快。这当然是个好消息。但问题在哪里呢？问题就在使用 mod_perl 时占用大量的内存。在一般情况下，CGI 脚本是以传统方式运行的，它提供服务后就消失。但在 mod_perl 中，为了将来能够使用，脚本常驻在内存中。这意味着在 mod_perl 下运行更多的脚本，它们将占用更多内存。

如果 Perl 脚本使用了大量的共用模块，可以通过一个简单的 Perl 脚本预装它们。例如，下面这个脚本 loader.pl 就可以预装 CGI.pm 模块：

```
#!/usr/local/bin/perl
#  startup script
use  lib qw(/www/perl/modules) ;
use  CGI;
1;
```

可以通过 PerlScript 命令预装这个共用模块：

```
PerlScript  /path/to/loader.pl
```

在大多数的现代操作系统中，运行 mod_perl 的 Apache 子进程可以共享 CGI.pm。

大家都知道，在任何情况下，装载的脚本越多，所需要的内存越多。如果只有有限的内存，并需要注意服务器的内存消耗，那么内存的使用将是个问题。利用通用的 UNIX 工具，例如 ps 或 top，就可以很容易地监视内存消耗。可在 Linux 系统上使用下面的命令产生与 httpd 有关的重要信息列表：

```
ps auxw | awk '{print $1, "\t", $2 , "\t", $5, "\t", $6} ' |
egrep  '( httpd | SIZE )'
```

它将打印所有 httpd 子进程的 USER、PID、SIZE（虚拟图像容量，文本、数据和堆栈的容量）、RSS（常驻设置容量，内存中的脚本千字节数）。

也可以利用 Apache::Status 模块发现当前 mod_perl 是如何将 Perl 软件包和脚本放在缓冲区的。但在使用 Apache::Status 模块前，需要安装 Devel::Symdump 模块。Devel::Symdump 模块提供了一种方便的方法来检查正在运行的程序的 Perl 符号表和类继承。如果已经安装了 CPAN.pm 模块，可以使用如下命令：

```
perl -MCPAN -e  'install "Devel::Symdump"'
```

或者从下面的 URL 得到源代码：

```
www.perl.com/cgi-bin/cpan_mod?module=Devel::Symdump
```

安装 Devel::Symdump 模块很容易。只要阅读模块中的 README 文件就知道怎样安装了。Devel::Symdump 模块安装好后，可以得到已被装入（在 Apache 子进程）的模块的

信息和模块的符号表。在你的一个 Apache 配置文件中定义如下的配置段，并重启 Apache 服务器：

```
<Location  /perl-status>
SetHandler  perl-script
PerlHandler  Apache::Status
</Location>
```

就可以使用如下的 URL 得到状况信息（如图 13-1 所示）：

```
http: //yourserver/perl-status
```

这是一种很好检测进程装入的内容的方法。

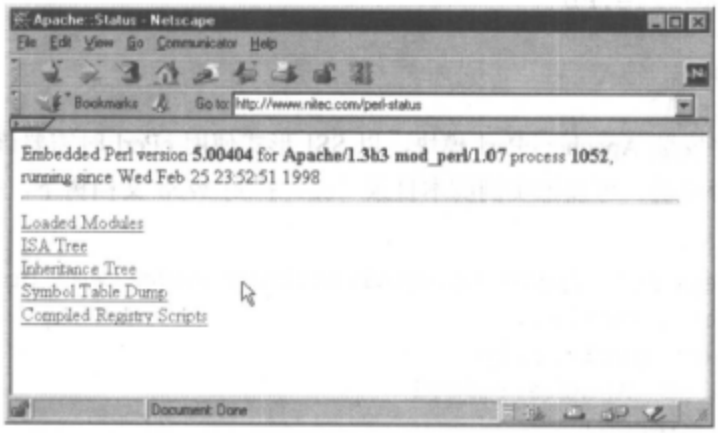


图 13-1　Perl 状态页

13.8.2　安全问题

由 mod_perl 运行的脚本的用户和组的权限与 httpd.conf 文件中的 User 和 Group 命令指定的用户和组的权限相同。需要确保这些用户/组的权限非常有限。

在开发脚本时需要注意在使用用户输入的数据前检查它。如果想在 mod_perl 中启动不安全检查，需要在 httpd.conf 文件中设置如下命令：

```
PerlTaintCheck On
```

常用的允许使用不安全检查的 -T 开关在 mod_perl 下不能使用。

13.9　其他用于 Apache 的嵌入 Perl

前面讨论了 mod_perl，现在提供一些在 Apache 中使用 Perl 的其他方法。在这些方法中，有两种值得一提：ePerl 和 embPerl。

根据 ePerl 的作者，Ralf S. Engelschall，的说法，ePerl 曾是第一个嵌入的 Perl 解释程序。ePerl 开始是一种独立程序，它可以解析和处理嵌入在简单 ASCII 文本文件中的 Perl 脚本。它常用作内容生成过滤程序，也是好的 SSI 候选语言。ePerl 程序是全功能的 Perl 解释程序，它使用原始的 Perl 解释程序库（libperl.a）。因此，ePerl 和 Perl 的语言是相同

的，对于 Perl 程序员而言，没有什么新东西要学。可以把 ePerl 与 Perl 使用的所有模块一起使用。这包括 DynaLoader 工具装入的共享对象。从下面这个 Web 站点可以自由下载 ePerl 软件：

```
www.engelschall.com/sw/eperl/
```

下载软件后，可以按照软件包提供的指示在你的系统上安装 ePerl。注意，如果想只以 SSI 形式使用 ePerl，就不需要安装独立的 ePerl。否则，需要在已有的 Perl 系统的 /path/to/perl/lib/site_perl/{Parse,Apache}/目录下安装集成的 Perl 5 模块。可以在 ePerl 源目录下运行下面的命令：

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

```
perl  Makefile.PL
make
make test
make install
```

这些命令将安装 Apache::ePerl 模块，以 SSI 形式使用 ePerl 时需要该模块。如果想快速测试安装是否成功，可以把实例脚本目录（eg）拷到 Web 文档树下，并修改 httpd.conf 如下：

```
Alias  /eperl/  /path/to/eg/directory under Web document tree/
<Location  /eperl/>
SetHandler  perl-script
PerlHandler  Apache::ePerl
Options  ExecCGI
</Location>
```

重启 Apache 并确保 Web 服务器可读实例脚本目录。现在就可以访问 ePerl 脚本的例子了。例如，ePerl 作者提供了的一个如列表 13-2 所示的例子。

列表 13-2：demo.env.phtml

```
<!-
##
##  demo.env - ePerl demonstration  webpage
##  Copyright (c) 1996,1997  Ralf  S. Engelschall,
All Rights Reserved.
##
->
<html>
<head>
<title>demo.env</title>
</head>
<body>
<blockquote>
<blockquote>
<h1>demo.env</h1>
<h2>Standard CGI Example: Enviroment</h2>
<p>
```

```
This prints out the CGI enviroment provided by the Web server
as a sorted list consisting of key/value pairs.

<p>
<pre>
<? My $key;
  foreach $key (sort(keys(%ENV))) {
    print "$key=$ENV{$key}\n";
  }
!>
</pre>

</blockquote>
</blockquote>
</body>
</html>
```

这段脚本将生成如图 13-2 所示的输出。

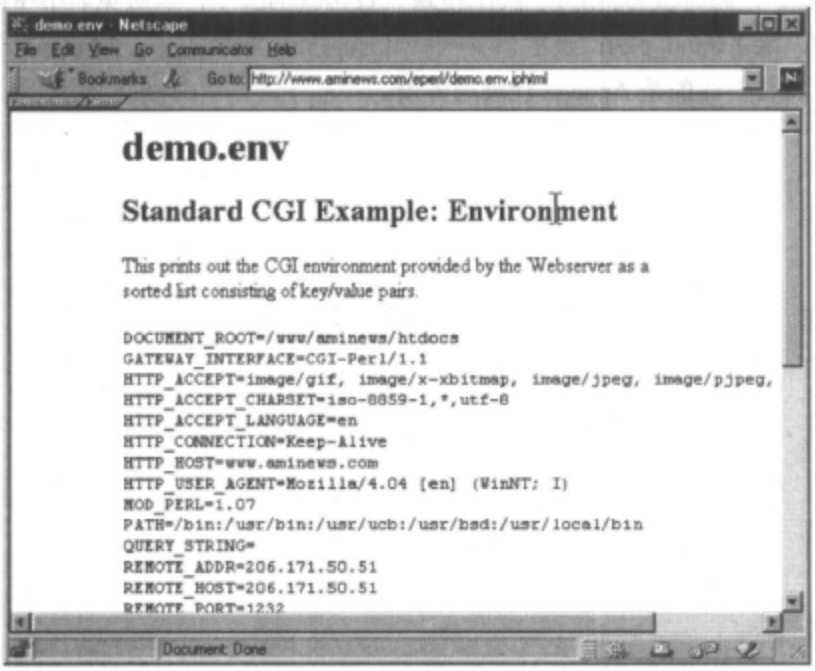


图 13-2　demo.env.iphtml 的输出

就像列表 13-2 中所看到的那样，真正的 Perl 代码嵌入在 HTML 文件中，并包含在如下的标记对中：

```
<?
  # perl code goes here
 !>
```

正如前面所说的，在基于 ePerl 的 SSI 页中可以写入任何有效的 Perl 代码，这些 Perl 代码可以提供大量的功能。有一个例子讲述 Net::FTP 模块怎样从远程 FTP 服务器中得到

文件并把它提供给页的访问者。还有许多其他有用的例子值得一看。

所喜欢的另一个嵌入 Perl 是 Gerald Richter 的 embPerl。从下面这个 FTP 站点可以下载 embPerl：

ftp://ftp.dev.ecos.de/pub/perl/embperl/

embPerl 和 ePerl 在很多方面都相似。它们真正的不同之处是 ePerl 不是 HTML 专用的，而 embPerl 是专用于 HTML 的。embPerl 提供了一些元命令来完成一些特殊的任务，例如填充 HTML 表等。它也以特殊的方式解释某些 HTML 标记。这一点在某些场合很有用，但学新语法可能麻烦了一点。如果需要用于一般目的的嵌入 Perl 方案，ePerl 更合适。

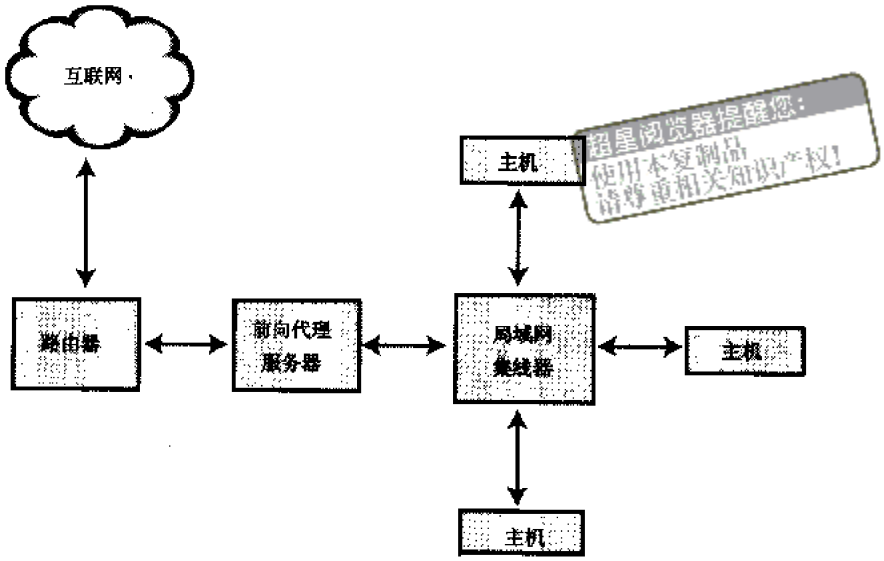
13.10 小　　结

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

在本章中，讨论了如何在 Apache 上通过 mod_perl 模块使用 Perl。mod_perl 模块提供了一种简单的移植方法，可以把传统的基于 Perl 的 CGI 脚本转换为高性能的基于 mod_perl 的脚本。由于一般程序员在写 CGI 脚本时并不是非常细心，所以建议用面向对象的 Perl 重写 CGI 脚本。同时极力推荐用 Perl 模块代替传统的脚本，以避免名字空间冲突并可节省系统资源。除了 mod_perl，还提供了一些其他的嵌入 Perl 方案，如 ePerl、embPerl。

下一章将要讲述的是 Apache 代理服务器。

减少带宽的使用和实现用户的访问策略。



超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

图 14-1 前向代理服务器

这类代理服务器也称为缓冲代理服务器。逆向服务器也可以缓冲数据，但它的作用恰好与前向代理服务器相反。

14.1.2 逆向代理服务器

逆向代理服务器位于互联网资源前面，或位于企业内联网资源前面，如图 14-2 所示。在这种设置中，逆向服务器从原始服务器找到被请求的资源，并把它返回给用户主机。

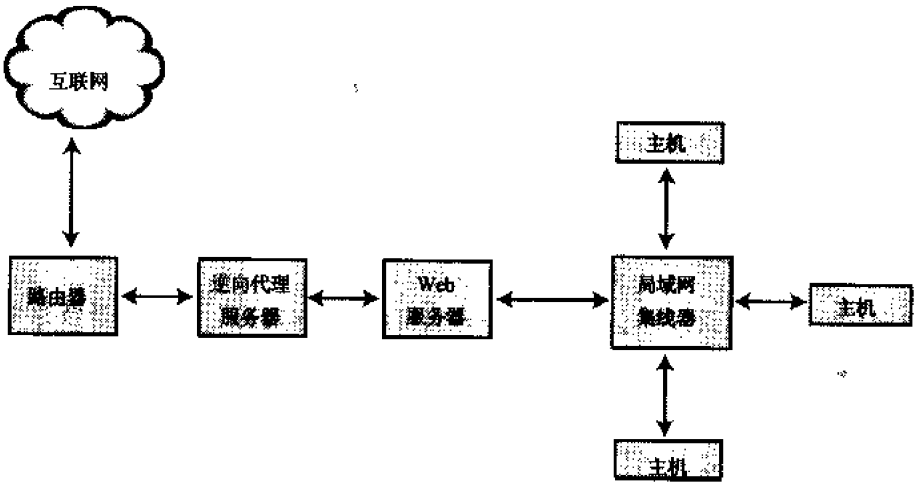


图 14-2 逆向代理服务器

与前向代理服务器不同的是，逆向代理服务器的用户主机并不知道它们连接的是代理服务器而不是资源服务器本身。从最终用户的角度看，它可以直接访问被请求的资源。

例如，如果有一个逆向代理服务器为 Web 站点 www.csus.edu 提供服务，那么所有的 CSUS 学生所要做的只是把他们的浏览器连向 www.csus.edu，而并不需要告诉浏览器任何关于代理服务器的信息。浏览器向称为 www.csus.edu 的服务器发请求。它并不知道 www.csus.edu 服务器实际上是一个前向代理服务器，这个前向代理服务器把请求内部传输给一个称为 internal-www.csus.edu 的 Web 服务器，得到所请求的资源。这种设置有什么好处呢？因为每一个请求的数据是缓冲的，所以代理服务器可以为后面的实际服务器提供平衡负载的支持。

Apache 现在不支持逆向代理服务，但 Apache 服务器的下一版本将实现逆向代理。

14.2　谁应当使用代理服务器

代理服务器对多个用户访问的网络是理想的。通常情况下，许多组织机构总有几台计算机通过单独的连接来访问互联网，例如通过 ISDN 路由器、专线连接、按需连接。代理服务器在这样的网络中非常有用。

在互联网和企业内联网中使用代理服务器有以下优点：

♦代理——如果内联网出于安全或费用的原因使用的是不可路由的 IP 地址，那么可以通过代理服务器向不能进入互联网的主机提供互联网资源。本章中将讨论如何实现这一点。

♦缓冲——使用像 Apache（带有 mod_perl）这样的缓冲代理服务器，可以使本地用户更快地访问互联网资源。这不仅提高了用户对于网络性能的感受，同时降低了带宽的使用费用。

♦登录和访问控制——使用代理服务器可以监视内部人员或学生使用互联网（甚至是企业内联网）的情况。可以阻止用户对某些特定 Web 站点的访问，保护你的公司避免潜在的诉讼，阻止滥用公司的时间。通过分析代理服务器的访问和错误记录可以确定网络的使用方式，为将来制定更好的网络使用方针。

就像所看到的那样，使用代理服务器有许多益处。现在来看一看如何把 Apache 转换为代理服务器。

14.3　Apache 用作代理服务器

在 Apache 中对代理服务器的支持来自 mod_perl 模块。在默认情况下，这个模块并不被编译。可能需要在 Apache 源目录下将这一模块增加到配置文件中，并重新编译 Apache。现在，Apache 只能实现缓冲代理服务器。它可以支持 HTTP（HTTP/0.9 和 HTTP/1.0）、HTTPS（通过 SSL 的 CONNECT）和 FTP 协议。mod_perl 模块也可以被配置为连接到其他代理模块。它提供下面这些命令。

14.3.1 ProxyRequests

语　　法：ProxyRequests on/off
缺 省 值：ProxyRequests Off
使用范围：服务器配置，虚拟主机
这条命令允许使用或禁止缓冲代理服务。但它不影响 ProxyPass 命令的功能。

超星阅览器提醒您：
使用本复制品
请遵守相关知识产权！

14.3.2 ProxyRemote

语　　法：ProxyRemote <match> <remote-proxy-server-URL>
缺 省 值：无
使用范围：服务器配置，虚拟主机
这个命令允许你的代理服务器连接其他代理服务器。match 值可以是下面的值之一：

- 远程服务器所支持的 URL 模式的名称。
- 远程服务器使用的部分 URL。
- 指示对所有请求连接的服务器。

<remove-proxy-server-URL>可以是 http://remove-proxy-hostname:port。注意，当前只支持 HTTP 协议。也就是说，只能定义处理 HTTP 协议的代理服务器。但可把 FTP 请求从你的代理服务器传给某个支持 HTTP 和 FTP 协议的服务器，例如：

```
ProxyRemote ftp http://ftp.proxy.nitec.com:8000
```

这个命令将把来自本地代理服务器的所有 FTP 请求发送给 ftp://ftp.proxy.nitec.com。这些请求也可以通过 HTTP 发送，因此实际的 FTP 传输发生在远程代理服务器上。

如果只想把某个 Web 站点的所有代理请求直接发送给代理服务器，就可以用 ProxyRemote 命令，例如：

```
ProxyRemote http://www.bigisp.com/   http://web-proxy
.bigisp.com:8000
```

它将把与 www.bigisp.com 匹配的所有请求发送给 web-proxy.bigisp.com。如果想把你的代理服务器的所有请求传送给其他代理，可以使用作为通配符的星号，例如：

```
ProxyRemote * http://proxy.domain.com
```

它将把所有本地代理请求发送给在 proxy.domain.com 的代理服务器。

14.3.3 ProxyPass

语　　法：ProxyPass <relative-URL> <destination-URL>
缺 省 值：无
使用范围：服务器配置，虚拟主机
这个命令可以在代理服务器的文档空间映射 Web 服务器文档树。例如：

```
ProxyPass  /internet/microsoft   www.microft.com/
```

ProxyPass 命令可以在代理服务器 proxy.nitec.com 的 httpd.conf 文件中找到，它允许代理服务器的用户通过下面 URL 访问微软 Web 站点：

```
http://proxy.nitec.com/internet/microsoft
```

它的作用如同远程 Web 站点的镜像。任何使用<relative-URL>的请求都被内部转换成<destination-URL>的代理请求。

Note　如果远程站点包含绝对引用，图像将不被显示，连接也不会工作。
同时要注意，现在不能对 SSL 目的服务器使用这个命令。

超星阅览器提醒您：

14.3.4　ProxyBlock

语　法：ProxyBlock <partial or full host name> ...
缺 省 值：无
使用范围：服务器配置，虚拟主机
这个命令允许阻塞对指名的主机或域的访问。例如：

```
ProxyBlock  gates
```

这个命令不允许访问名称中有 gates 的任何主机。因此，对 http://gates.ms.com 或 http://gates.friendsofbill.com 的访问将被阻塞。也可以像下面这样说明多个主机：

```
ProxyBlock apple orange.com bannana.com
```

它将阻塞所有与上面的字或域名匹配的主机的访问。mod_proxy 模块在服务器启动时确定这些主机的 IP 地址并把它们存起来，用于以后的匹配。

可以用下面的命令阻塞所有的主机访问：

```
ProxyBlock  *
```

它等效于禁止你的代理服务器的工作。

14.3.5　NoProxy

语　法：NoProxy <Domain name| Subnet | IP Address | Hostname>
缺 省 值：无
使用范围：服务器配置，虚拟主机
这个命令允许在企业内联网环境中控制 ProxyRemote 命令。可以指定域名、子网、IP 地址或主机名，它们将不由 ProxyRemote 命令中定义的代理服务器提供服务。例如：

```
ProxyRemote  *  http://firewall.yourcompany.com:8080
NoProxy         .yourcompany.com
```

这里，所有对<anything>.yourcompany.com 的请求（例如 www.yourcompany.com）都由本地代理服务器提供服务，其他的请求都送到 firewall.yourcompany.com 代理服务器。

14.3.6　ProxyDomain

语　法：ProxyDomain <Domain>
缺 省 值：无
使用范围：服务器配置，虚拟主机

这个命令为代理服务器定义缺省域名。当这个命令设置为企业内联网上的本地域名时，任何不包含域名的请求都将附加这个域名。例如：

```
ProxyDomain   .nitec.com
```

超星阅览器提醒您：
此为本复制品
请尊重知识产权！

当 nitec.com 域的用户发送如 http://marketing/us.html 的 URL 请求时，这一请求将被重新生成为如下的 URL：

http://marketing.nitec.com/us.html 请求

注意定义的域名前面必须有一个句点。

14.3.7 CacheRoot

语　　法：CacheRoot <directory>

缺 省 值：无

使用范围：服务器配置，虚拟主机

这个命令允许使用硬盘进行缓冲。可以为代理服务器指定写缓冲文件的目录名。运行代理模块的 Apache 服务器必须具有该目录的写权限，例如：

```
CacheRoot /www/proxy/cache
```

这个命令告诉 Apache 把代理缓冲数据写在/www/proxy/cache 目录下。注意在代理服务器使用目录前要用 CacheSize 命令定义缓冲区的大小。还需要使用其他的缓冲区命令（稍后讨论）来创建可用的硬盘缓冲代理方案。

14.3.8 CatcheSize

语　　法：CatcheSize <n kilobytes>

缺 省 值：CacheSize 5

使用范围：服务器配置，虚拟主机

这个命令定义了用做硬盘缓冲的硬盘空间（以 K 字节为单位）。缓冲文件写在 CacheRoot 命令定义的目录下。注意，对于代理服务器来说，写下比规定的空间更多的信息是不可能的，因为代理服务器的垃圾收集机制将删除文件，直到信息所占的空间小于或等于规定的空间。缺省设置是 5K，它是不现实的。建议根据用户负载把 CatcheSize 命令设置为 10MB 到 1GB 之间。

14.3.9 CatcheGcInterval

语　　法：CacheGcInterval <n hour>

缺 省 值：无

使用范围：服务器配置，虚拟主机

这个命令定义的是 Apache 为了能够及时删除缓冲区中过期的文件而检查缓冲区目录的时间间隔（以小时为单位）。这也是 Apache 检查由 CacheSize 命令定义的硬盘空间的使用情况的时间间隔。

14.3.10 CacheMaxExpire

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

语　　法：CacheMaxExpire <n hour >

缺 省 值：CacheMaxExpire 24

使用范围：服务器配置，虚拟主机

这个命令定义的是所有缓冲文档过期的时间（以小时为单位）。也就是说，如果像下面这样定义 CacheMaxExpire 命令：

```
CacheMaxExpire 48
```

那么所有的缓冲文档将在 48 小时或两天内过期。这个命令将覆盖文档中定义的任何过期时间。因此，即使某个文档的过期时间比这个命令定义的最大时间还长，该文档仍然被删除。

14.3.11 CacheLastModifiedFactor

语　　法：CacheLastModifiedFactor <floating point number>

缺 省 值：CacheLastModifiedFactor 0.1

使用范围：服务器配置，虚拟主机

这个命令定义了一个系数，如果原始 Web 站点不提供文档的过期时间，那么该系数用于计算过期时间。计算公式如下所示：

```
expiry-period = (last modification time for the document)
* (floating point number)
```

因此，如果文档最近一次是 24 小时前被修改的，那么默认系数 0.1 表示 Apache 计算出的该文档的过期时间是 2.4 小时。如果被算出的过期时间比 CacheMaxExpire 设置的时间长，那么后者优先。

14.3.12 CacheDirLength

语　　法：CacheDirLength <length>

缺 省 值：CacheDirLength 1

使用范围：服务器配置， 虚拟主机

如果系统使用硬盘缓冲，那么 Apache 将在 CacheRoot 命令定义的目录下创建子目录。CacheDirLength 命令说明的是子目录名的字符数。实际上不用改变它的默认值。如果想知道这些子目录如何创建的以及为什么被创建，下面就是答案。

Apache 为一个被缓冲的 URL 数据创建路径和文件名时使用 HASH 散列法。例如，如果要通过代理 Apache 服务器访问一个 URL（例如 www.microsoft.com），服务器将散列这个 URL，这样就可以很快找到数据。这种散列法看起来像 1YSRxSmB20Q_HkqkTuXeqvw。如果 CacheDirLength 和 CacheDirLevels 命令都使用默认值，那么 Apache 会把基于 www.microsoft.com 的数据存入下面的文件：

```
%CacheRoot%/1/Y/S/RRxSmB20Q_HkqkTuXeqvw
```

这里%CacheRoot%是 CacheRoot 命令说明的目录。1/Y/S 是根据 CacheDirLevels 命令

的默认值生成的。

当使用相同的 URL 再次请求这个文档时，Apache 将重新进行散列计算，并从指定的路径取得数据。

14.3.13 CacheDirLevels

语　　法：CacheDirLevels <levels>
缺 省 值：CacheDirLevels 3
使用范围：服务器配置，虚拟主机

这个命令说明的是 Apache 将为缓冲文件创建的子目录的层数。相关信息请参阅“CacheDirLength”部分。

超星阅览器提醒您：

14.3.14 CacheDefaultExpire

语　　法：CacheDefaultExpire <n hour>
缺 省 值：CacheDefaultExpire 1
使用范围：服务器配置，虚拟主机

这个命令说明的是一个默认时间（以小时为单位），它是当无法确定缓冲文件的最近修改时间时的过期时间的默认值。CacheMaxExpire 不覆盖这个值。

14.3.15 NoCache

语　　法：NoCache <Domain name | Subnet | IP Address | Hostname> ...
缺 省 值：无
使用范围：服务器配置，虚拟主机

NoCache 命令定义的是由空格隔开的主机、域名、IP 地址列表，它们都不实现缓冲。在企业内联网中，该命令可用于禁止本地 Web 站点的缓冲。注意代理服务器也匹配主机部分名。如果想禁止所有的缓冲，用下面的命令：

```
NoCache *
```

14.4 配置代理服务器

现在已经了解了可以用哪些命令配置 Apache 中的代理，下面来看一看配置的细节。

为了允许使用代理服务器，需要把 ProxyRequests 设为 On。根据你希望代理服务器做些什么进行附加配置。无论希望做什么，你所选的代理配置都应该放入一个特殊的 <Directory>容器中。

```
<Directory proxy: *>
...
</Directory>
```

任何用于控制代理服务器的命令都应该放这个容器中。星号代表被请求的 URL 通配

符。也就是说，如果由 Apache 服务器处理 www.nitec.com 的请求，<Directory>容器看起来是这样的：

```
<Directory proxy:http://www.nitec.com >
...
</Directory>
```

Note

也可以使用<Directory~/RE/>容器，它使用的是规则表达式，例如：

```
<Directory ~ proxy:http://[^:/]+/.*>
...
</Directory>
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

下面讲述的是一些常用的代理配置。

14.4.1　将私有 IP 网连到互联网

假设私有网上只有一台计算机被分配了可以由互联网路由的 IP 地址，如图 14-3 所示。这台计算机运行 Apache 代理服务器，ProxyRequest 设为 on，并且不需要附加的代理配置。所有请求由代理服务器进行服务。

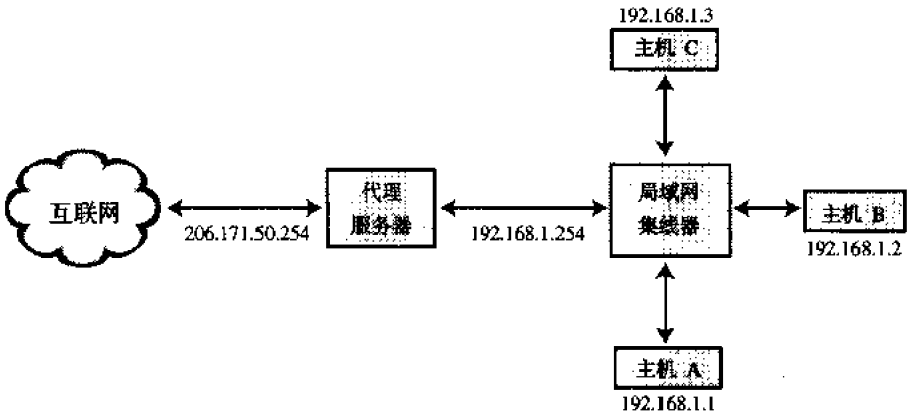


图 14-3　将私有 IP 网连到互联网的代理服务器

注意，在这样的配置中，代理服务器应该是多地址的。也就是说，它既要能访问不可路由的私有网络（192.168.1.0）又要能访问可路由的 IP 网络（206.171.50.0）。对于私有网而言，代理服务器的作用类似于防火墙。代理服务器允许主机访问互联网设备，例如 Web、FTP 等。

现在讨论缓冲代理服务器。

14.4.2　远程 Web 站点的缓冲存信者

由于互联网和企业内联网的大量内容都是静态的，所以把这些内容缓冲存储在本地服务器上可以节省可观的网络带宽。一个允许使用缓冲存储的代理服务器只在缓冲区中没有被请求的文档或缓冲区中被请求的文档已过期时，才去服务器取请求的文档。如果想把你

的代理服务器设为允许进行缓冲存储，需要在一个特定的目录容器中指定缓冲命令，例如：

```
<Directory  proxy:* >
CacheRoot  /www/cache
CacheSize  1024
CacheMaxExpire  24
</Directory>
```

这个配置定义了向/www/cache 目录写缓冲文件的缓冲代理服务器。它允许写 1024K（1MB）的数据，每 24 小时后缓冲区将过期。

如果不希望外面的用户滥用你的代理，可以通过主机认证或用户名/口令认证来限制对代理的访问。

超星阅览器提醒您
使用本复制品
请尊重相关知识产权！

如果想控制访问代理服务器的主机，可以创建一个如下的配置：

```
<Directory proxy:*>

AuthType  Basic
AuthName  Proxy
Order deny, allow
deny from all
allow from myhost.nitec.com

</Directory>
```

这个配置只允许 myhost.nitec.com 访问。如果要使用用户名/口令认证，可以采用类似下例的方法：

```
<Directory proxy:*>

AuthType  Basic
AuthName  Proxy
AuthUserFile  /path/to/proxy/.htpasswd
AuthName  Proxy
Require valid-user
</Directory>
```

如果不清楚怎样创建必要的口令文件，可以参阅第 10 章的内容。

也可以限制某种协议的访问，例如：

```
<Directory proxy:http: * >
...
</Directory>
```

它允许控制你的代理服务器如何处理 HTTP 请求。类似的，也可以控制代理服务器如何处理其他的协议。

```
<Directory proxy:ftp:*>
...
</Directory>
```

```
<Directory proxy:https:*>
...
</Directory>
```

也可以为代理服务器创建单独的虚拟主机。那样，应该在代理主机的<VirtualHost>容器中指定命令。

```
<VirtualHost proxy.host.com:*>
...
 </VirtualHost>
```

14.4.3 建立 Web 站点的镜像

Web 站点的镜像是远程 Web 站点的本地拷贝。例如，如果现在希望建立 www.apache.org 的 Web 站点的镜像，这样你的用户就可以连接镜像站点并快速访问 Apache 的信息，那么可以使用代理服务器创建如下的镜像：

```
ProxyPass  /  www.apache.org /
CacheRoot  /www/cache
CacheDefaultExpire 24
```

这将使代理服务器建立 www.apache.org 的 Web 站点的镜像。例如，这种配置可以把代理服务器 blackhole.nitec.com 转变为如图 14-4 所示的 Apache 镜像。

图 14-4　用代理服务器建立 Apache Web 站点的镜像

如果用户的 URL 是 http://blackhole.nitec.com，那么他将看到 Apache 镜像的索引页，就像访问 www.apache.org 一样。

在建立其他人的 Web 站点的镜像前，得到对方的许可是很重要的，因为可能会涉及到版权问题。

14.5 设置 Web 浏览器

现在已经可以在客户机上设置 Web 浏览器了。流行的 Web 浏览器非常容易使用代理服务器。下面将要讲述的是如何为代理配置最新版本的 Netscape Navigator 和 Microsoft Internet Explorer（IE）。有两种方法可以为这些浏览器创建代理服务器：手工代理配置和自动代理配置。

如果你的客户较少并且代理配置不常变，那么应该采用手工配置。否则，可以跳到“Web 浏览器的自动代理配置”段。

下面的步骤将引导进行手工代理配置：

1. 选中 Navigator 菜单条的 Edit 下的 Preference 选项。可以看到如图 14-5 的对话窗口。

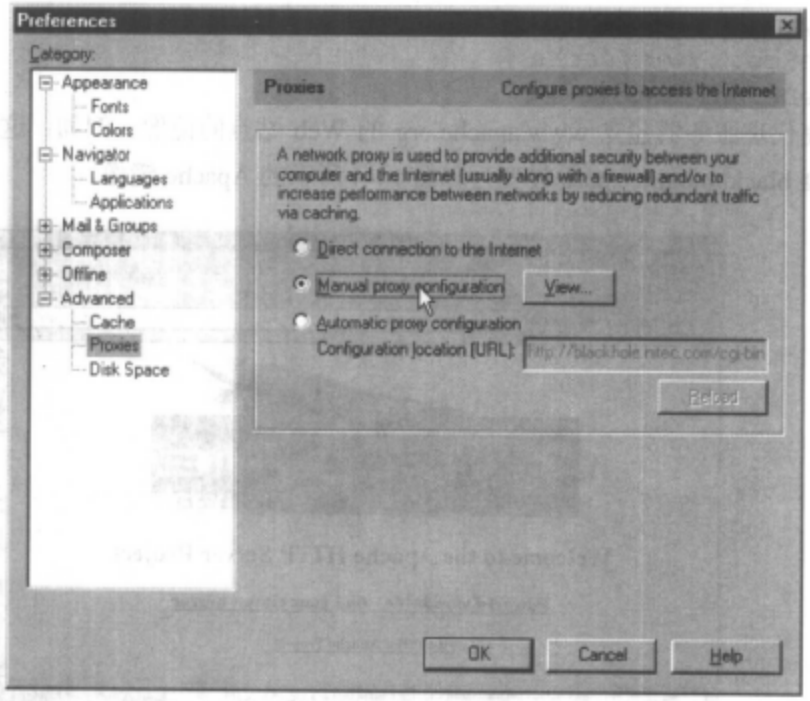


图 14-5 Netscape Navigator Preference 窗口

2. 单击高级类。

3. 单击代理类。

4. 选中手工代理配置按钮，单击视图按钮。可以看到如图 14-6 所示的窗口。

5. 录入关于 HTTP、FTP、安全性（HTTPS）的代理服务器使用的 URL 和端口信息。由于使用代理服务器处理所有的协议，所以 URL（http://proxy.nitec.com/）和端口（8080）都是一样的。如果由不同的代理服务器处理不同的协议，应该输入相应的内容。

6. 录入信息后，发一个访问远程文档的请求，检查你的代理是否工作正常。有一个很好的检测方法，就是监视代理服务器的访问和错误日志。大多数 UNIX 系统都可以使用

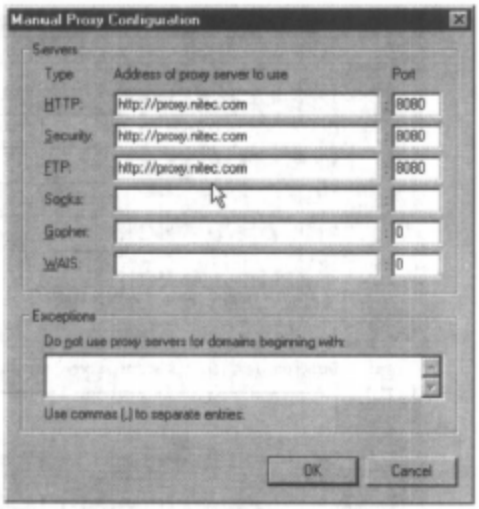


图 14-6　Netscape Navigator 手工代理设置窗口

下面的命令来观察写入文件的日志登录：

```
tail -f /path/to/access/log
```

现在配置其他的 Web 浏览器。为 Internet Explorer 进行手工代理配置的步骤如下：

1. 单击视图菜单项，选中互联网选项，将弹出如图 14-7 的对话窗口。

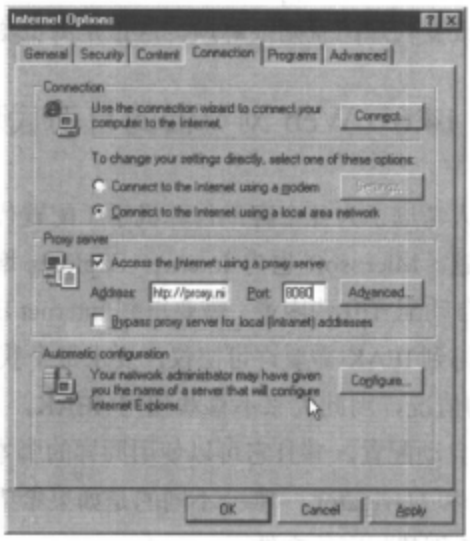


图 14-7　Microsoft Internet Explorer 手工代理设置窗口

2. 选中连接表页。
3. 选中使用代理服务器访问互联网按纽。
4. 输入代理服务器的 URL 和端口号。
5. 单击 OK 完成配置。

如果想为不同的协议定义不同的代理服务器，点按高级按纽，将弹出如图 14-8 所示的窗口。

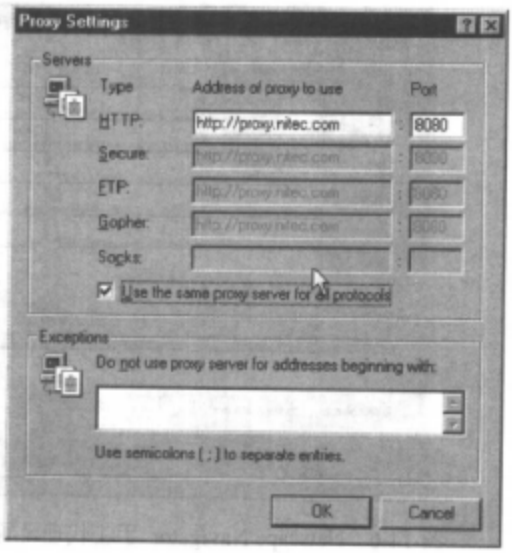


图 14-8 Microsoft Internet Explorer 高级代理设置窗口

在这里，就像在 Netscape Navigator 中一样，可以定义不同的代理服务器设置。一旦单击 OK 并用 Apply 按钮来增加新设置，你的浏览器就配置为使用代理服务器了。

就像所看到的那样，为代理手工配置 Web 浏览器并不复杂。然而，如果有许多用户计算机需要配置，每次改变代理配置则很麻烦。这正是浏览器自动代理配置方便之处。

14.6 Web 浏览器的自动代理配置

Netscape 中的好心人考虑了为几个客户计算机手工配置代理所引入的问题，并提供了一种方法来解决此类问题。Microsoft 也为 Internet Explorer 增加了自动配置选项。不幸的是，在浏览器上的实现自动配置有些困难。需要得到 Internet Explorer 管理工具包（IEAK）来创建自动配置文件。得到 IEAK 需要许可协议，要求每个获得许可的人每季向 Microsoft 报道关于 IEAK 使用的情况，因此大家不喜欢使用 IEAK。从可靠信息来源获悉：IEAK 文档有类似 Netscape 的自动配置，并且它可以使用同样的脚本。

本段内容适用于 IE 和 Navigator。唯一不同的是如果希望在 IE 上实现自动代理配置，需要解决如何使用 IEAK 创建相应的文件。

代理自动配置使用的是特殊的 JavaScript。对于 Netscape Navigator 和 IE 都是这样。特殊的 JavaScript 有以下要求。

代理自动配置 JavaScript 必须执行称为 FindProxyForURL 的函数。这个函数具有如下的框架：

```
function FindProxyForURL(url, host)
  {
    // java script code goes here
    return "proxy to use for servicing the URL";
```

```
}
```

这个函数接收的参数是 url 和 host。url 是被请求的完整的 URL，host 是从 URL 提取的主机名。例如，假设有一个请求是 Web 浏览器发出的探测，它将调用下面的函数：

```
ret = FindProxyForURL(", )
```

Note　函数调用中的主机参数是://与第一个:或第一个/之间的子串。端口号不包含在这个参数中。

该函数的返回值是字符串，其中含有对特定的 URL 请求的必要的代理配置。下面是一些表示代理配置的可接受字符串值：

NULL：如果返回 NULL 值（不是字符串 NULL），它告诉浏览器对该请求不使用任何代理。

浏览器提醒您：使用本复制品请尊重相关知识产权！

DIRECT：可以直接连接，不需要任何代理。

PROXY host:port;：应该使用指定的代理。

SOCKS host:port;：应该使用指定的 SOCKS 服务器。

很明显，真正有意义的返回值是 DIRECT 和 PROXY。当具有多个代理或 SOCKS 服务器时，可以返回一串 host:port 对。例如：

```
PROXY best-proxy.nitec.com:8080; PROXY good-proxy
.nitec.com:8081; PROXY soso-proxy.nitec.com:8082
```

上面这个代理配置告诉浏览器先试 best-proxy.nitec.com，如果失败，再试下一个（good-proxy.nitec.com）等等。注意，每个 host:port 对都是由分号分隔的，每一对都有关键字 PROXY。如果所有的代理服务都失败了，将会出现提示信息，让用户选择是否进行直接连接。

Note　如果所有的代理都失败了，并且没有 DIRECT 选项，浏览器将询问用户是否暂时忽略代理并进行直接连接。

为了避免用户的交互作用，可以用以下内容代替上面的配置：

```
PROXY best-proxy.nitec.com:8080; PROXY good- proxy
.nitec.com:8081; PROXY soso-proxy.nitec.com:8082; DIRECT
```

这里，直接连接已列为最后选项，所以如果代理失败，系统将不提示用户就进行直接连接。也可以混合使用 PROXY 和 SOCKS。例如：

```
PROXY best-proxy.nitec.com:8030; SOCKS socks4
.nitec.com:1080; DIREC
```

当主代理服务器 best-proxy.nitec.com 响应失败时，将使用基于 SOCKS 的代理。

如果一个代理响应失败，30 分钟后 Netscape Navigator 将重试代理。每次相继的查询失败后，重试间隔都将延长 30 分钟。

Web 管理者（他必须了解 JavaScript 脚本）可以使用一系列预定义函数完成他的工作。这些函数和它们的描述列在表 14-1 中。

表 14-1 用于编写自动代理配置脚本的预定义函数

函数名	说明	例子
IsPlainHostName(host)	如果 host 中没有点（也就是说，如果不包括域名）则返回 true	isPlainHostName("blackhole") 返回 true isPlainHostName("blackhole .nitec.com") 返回 flase
dnsDomainIs(host, domain)	如果 host 属于 domain，返回 true。注意 domain 名的开头是句点	dnsDomainIs("www.nitec.com", ".nitec.com") 返回 true dnsDomainIs("www.apache.org ", ".nitec.com") 返回 flase
localHostOrDomaiIs (host,fqdnhost)	如果 fqdnhost（完整的主机名）的主机部分与 host 匹配，返回 true	localHostOrDomainIs("a.b.com" , "a.b.com") 返回 ture localHostOrDomainIs("a.b", "a.b.com") 返回 true localHostOrDomainIs("a.b.org", "a.c.com")返回 false
isResolvable(host)	如果 DNS 服务器可以把主机名映射为 IP，就返回 true；否则 false。注意这个函数的使用会降低浏览器的速度，因为 DNS 请求需要进行测试	isResolvable("{hyperlink}"); 因为{hyperlink}有 DNS 记录，所以返回 ture
isInNet(host,IP address pattern, netmask)	如果主机的 IP 地址与第二个参数 IP address 说明的模式匹配，返回 true。匹配是通过网络掩码来实现的：如果掩码的某个八进制数是 255，匹配后的值不变；如果掩码的某个八进制数是 0，IP 地址匹配后该部分值被忽略。注意这个函数的使用会降低浏览器的速度，因为 DNS 请求需要进行测试	如果主机地址是 206.171.50.51;isInNet(host,"206.171.50.50", "255.255.255.0"); 返回 true。因为根据掩码可知只要前三部分相同即可，最后一部分可以忽略
dnsResolve(host)	如果成功将返回主机的 IP 地址。注意这个函数的使用会降低浏览器的速度，因为 DNS 请求需要进行测试	dnsReslove("proxy.nitec.com") 返回"206.171.50.50"
myIpAddress()	返回正在运行 Web 浏览器的主机的 IP 地址。注意这个函数的使用会降低浏览器的速度，因为 DNS 请求需要进行测试	var hostIP = myIpAddress()返回 Web 浏览器主机的 IP 地址并把它放在 hostip 变量中
dnsDomainLevels(host)	返回主机名中的域的层数	dnsDomainLevels("www.nitec. com") 返回 2

续表 14-1

函数名	说明	例子
shExpMatch(string, shellExpression)	如果字符串与 shell 表达式匹配，返回 true	shExpMatch("path/to/dir","*/to/*") 返回 true shExpMatch("abcdef","123") 返回 flase
weekdayRange(weekday1, weekday2, gmt)	只有第一个参数是必需的。如果这个函数计算出的日期与第一个参数相同，或者在第一个参数和第二个参数之间，则返回 true。如果第三个参数 gmt 是“GMT”，则用 GMT 时间代替本地时间 Weekday1 和 weekday2 可使用的值是：SUN、MON、TUE、WED、THU、FRI、SAT	weekdayRange("FRI")如果机器时间是星期五则返回 true weekdayRange("MON","FRI", "GMT") 如果 GMT 时间是在星期一到星期五之间，那么返回 true
dateRange(day) dateRange(day1, day2) dateRange(month) dateRange(month1, month2) dateRange(year) dateRange(year1, year2) dateRange(day1,month1, day2,month2) dateRange(month1,year1, month2,year2) dateRange(day1,month1, year1,day2,month2,year2) dateRange(day1,month1, year1,day2,month2,year2, gmt)	如果当前日、月、年或它们三个都在范围内，则返回 true。日的值是 1-31，月可以是 JAN、FEB、MAR、APRMAY、JUN、JUL、AUG、SEP、OCT、NOV、DEC。年是四位数字。gmt 是“GMT”或空（本地时间）	dateRange(31) 如果今天是第 31 日，则返回 ture dateRange("JAN,"APR") 如果这个月在一月和四月之间，则返回 true dateRange(1995) 如果今年是 1995 年，则返回 true
timeRange(hour) timeRange(hour1, hour2) timeRange(hour1, min1, hour2, min2) timeRange(hour1, min1, sec1, hour2, min2, sec2) timeRange(hour1, min1, sec1, hour2, min2, sec2, gmt)	如果指定的小时、分钟、秒是当前时间，则返回 true。如果当前相应时间单位在指定范围内，则返回 true。小时值是 0-23，分钟是 0-59，秒是 0-59，gmt 可以是“GMT”或空（本地时间）	timeRange(9,17) 如果当前时间是在上午 9 点到下午 5 点之间，返回 true

有了这些预定义函数和自定义函数，就可以写 FindProxyForURL 了，这样对每个请求都将返回相应的代理配置字符串。

当 Web 浏览器启动时，将从 JavaScript 文件（后面将讨论如何使 Web 浏览器使用这

将返回 false。接着调用 dnsDomainIs 函数看它是否属于.nitec.com 域。如果不是，也返回 false。由于这两个检查都返回 false，接着执行条件语句的 else 部分。Web 浏览器为这一 URL 请求返回如下的代理配置：

```
PROXY  proxy.nitec.com:8081;  DIRECT
```

这表示 Web 浏览器可以使用端口 8081 上的名为 proxy.nitec.com 的代理服务器（如果该服务器可以使用）。如果代理服务器不能使用，该请求由直接连接（direct）提供服务。对于大多数代理服务器，这个配置已经足够了。下面看一看更复杂的情况。

14.6.2　使用多个代理服务器

在这种情况下，假设有多个代理服务器。图 14-10 描述的是有三个代理服务器的网络。http-proxy.nitec.com 用于处理所有的远程 HTTP URL 请求；ftp-proxy.nitec.com 用于所有的远程 FTP URL 请求，ssl-proxy.nitec.com 用于所有的 HTTPS URL 请求。别的使用其他协议（例如 GOPHER、NEWS 等等）的请求将直接连接。所有种类的本地请求也都直接连接。

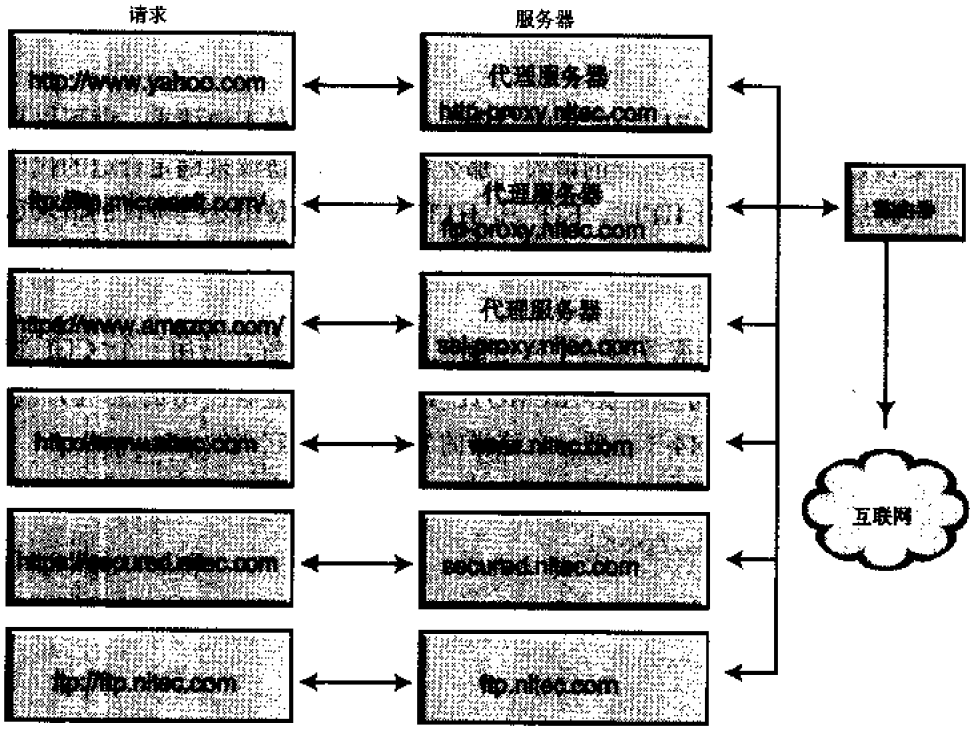


图 14-10　使用多个代理服务器

为了实现这个配置，FindProxyForURL 写得复杂一些，如列表 14-2 所示。

列表 14-2：多代理服务器配置的 FindProxyForURL

```
function  FindProxyForURL(url, host)  {
  // is the URL local? If it is then use a DIRECT connection
```

```
    if (isPlainHostName(host) ||
        dnsDomainIs(host, ".nitec.com"))  {
  return "DIRECT" ;
  }

// Okay, the URL is remote so check which proxy to use.
else {
if (url.substring(0,5 ) == "http:")  {
     return "PROXY http-proxy.nitec.com:8080";
 }
else if (url.substring(0,4) == "ftp:" )  {
   return "PROXY ftp-proxy.nitec.com:8080";
   }
else if (url.substring(0,6) == "https:" )  {
   return "PROXY ssl-proxy.nitec.com:8080";
   }
else {
     return "DIRECT";
   }
    }
}
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

这个函数首先检查 URL 请求是否是本地请求。如果是本地请求，就可以直接提供服务。如果该请求是对远程服务器的请求，那么使用 URL 协议来确定合适的代理服务器。但是，代理服务器只识别 HTTP、FTP 和 HTTPS 协议，使用这些协议的请求远程资源的 URL 分别由相应的代理服务器提供服务。如果远程 URL 请求不在上述协议范围内，它可以直接连接。

也可以根据访问代理服务器的主机定制代理服务器配置。使用一个 CGI 脚本就可以实现。这个 CGI 脚本可以根据 REMOTE_HOST（浏览器使用的主机）输出不同的 FindProxyForURL。列表 14-3 就是一个这样的 Perl 脚本，它叫做 proxy.pl。

列表 14-3：proxy.pl 脚本

```
#!/usr/local/bin/perl
#
# A perl script that outputs proxy server configuration.
# $Author$
# $Revision$
# $Id$

# Get the remote host IP from the CGI enviroment variable
# REMOTE_HOST
my $client = $ENV{REMOTE_HOST};

# Print out the necessary content-type to let the brower
# know that this is a proxy configuration.
```

```
print "Content-type:  application/x-ns-proxy-autoconfig\n\n";

# If the request came from a host with IP address
# 206.171.50.51 then output proxy configuration
# from subroutine &specialClient
#
if  ($client =~ /206\.171\.50\.51)  { &specialClient; }

# If the request came from any other clients, then
# output proxy configuration from the subroutine
# &otherClients
else { &otherClient; }

exit 0;

sub specialClient{
#
# This subroutine outputs a proxy server configuration
#

print <<FUNC ;

 function FindProxyForURL (url, host)
 {
  if (isPlainHostName(host) ||
     dnsDomainIs(host, ".nitec.com"))
     return "DIRECT";
 else if (shExpMatch(host, "*.com"))
return "PROXY com-proxy.nitec.com:8080;"

else if (shExpMatch(host, "*.edu"))
return "PROXY edu-proxy.nitec.com:8080;"

else
  return "DIRECT";
}
FUNC
}

sub otherClient{
#
# This subroutine outputs a proxy server configuration
#
print <<FUNC;

   function FindProxyForURL(url, host)
```

超星阅览器提醒您:
使用本复制品
请尊重相关知识产权!

```
{
        return "DIRECT";
  }

FUNC

}
```

这个脚本输出的是 IP 地址为 206.171.50.51 的主机的特殊代理服务器配置。所有其他主机各有不同的配置。为了访问代理配置，可以把浏览器（Navigator）设置为指向这个脚本所在的目录。它和指定 .pac 文件一样进行工作。在这个例子中，要求浏览器请求 CGI 脚本而不是.pac 文件。但是，由于这一脚本送出 pac 文件的内容类型，浏览器对为什么它从 CGI 脚本得到代理服务器配置并没有反应。虽然它不能完成很多功能，但可以为复杂的代理服务器配置使用类似的脚本。

14.7 小　　结

本章讲述如何将 Apache 用作缓冲（前向）代理服务器。在出现网络性能瓶颈的场合使用这种服务器可以降低响应时间，节省网络带宽，也有助于减少通信费用。因为代理服务器常常用于有大量用户通信的网络，所以本章也讲述了客户软件配置的各个方面，包括自动代理配置。

下一章将讨论安全 Socket 层（SSL）。

第 15 章 Apache 的安全 Socket 层

本章内容:

- 理解互联网的安全需要
- 什么是安全 socket 层（SSL）
- 为什么 Apache 中不包含 SSL
- 如何使用自由 Apache-SSL
- 如何在 Stronghold 中实现 SSL

超星阅览器提醒您:

仅在几年前，互联网还处于它的原始状态——一个由科学家和工程师使用的全球网。然而，随着 Web 的出现，互联网正在成为每个人的网络。现在，看起来每个人和每件事都在互联网上。同时，互联网也是新的商业领域。成千上万的商人，无论大的小的，都建立了 Web 站点，向全世界的新客户打开大门。但客户们总是很小心，因为他们知道互联网上并不是所有的地方都是安全的。

为了消除对这个新领域的不安全感，商业 Web 站点必须在它们和它们的客户之间提供保证安全传输的方案。安全 Socket 层（SSL）就是这样的一种技术。本章将讨论在使用 Apache 服务器的 Web 站点上这种新技术如何在促进商务活动方面扮演重要的角色。

15.1 SSL 的基础：加密

当数据从互联网的一个站点传向另一个站点时，它将经过许多计算机，如路由器、网关和其他网络设备等。例如，当 www.nitec.com/ Web 站点的访问者在这个站点上找到的 HTML 表单中输入了他/她的信用卡号后，数据的传输路径很可能如图 15-1 所示。

就像所看到的那样，数据在传输过程中必须通过许多节点，因此它可能在这些节点上被某个人截获。虽然数据包是以很快的速度传输（通常是毫秒）的，但它仍然可能被截获。这就是在交换敏感数据时需要安全机制的原因。安全是通过加密实现的。

从技术的角度讲，加密是一种数学编码方案；它保证只有预期的接收者才能访问数据；它对偷窃者隐藏了数据。加密方案广泛应用于限制对资源的访问。例如，如果要登录到 UNIX 系统或 Windows NT 系统，那么你的口令通常以加密方式保存在服务器的机器上。大多数 UNIX 系统对用户的口令进行加密，并把它与存在/etc/passwd 文件中的已加密的口令比较。如果两者一致，就允许该用户访问被请求资源。下面是两种不同的加密方案。

15.1.1 对称加密

这种方案类似于平常使用的锁和钥匙。你用钥匙开你的汽车，并用同一把钥匙锁汽车。类似地，在对称加密中，同一把密钥既用于开也用于关。图 15-2 就是对称加密方案例子。

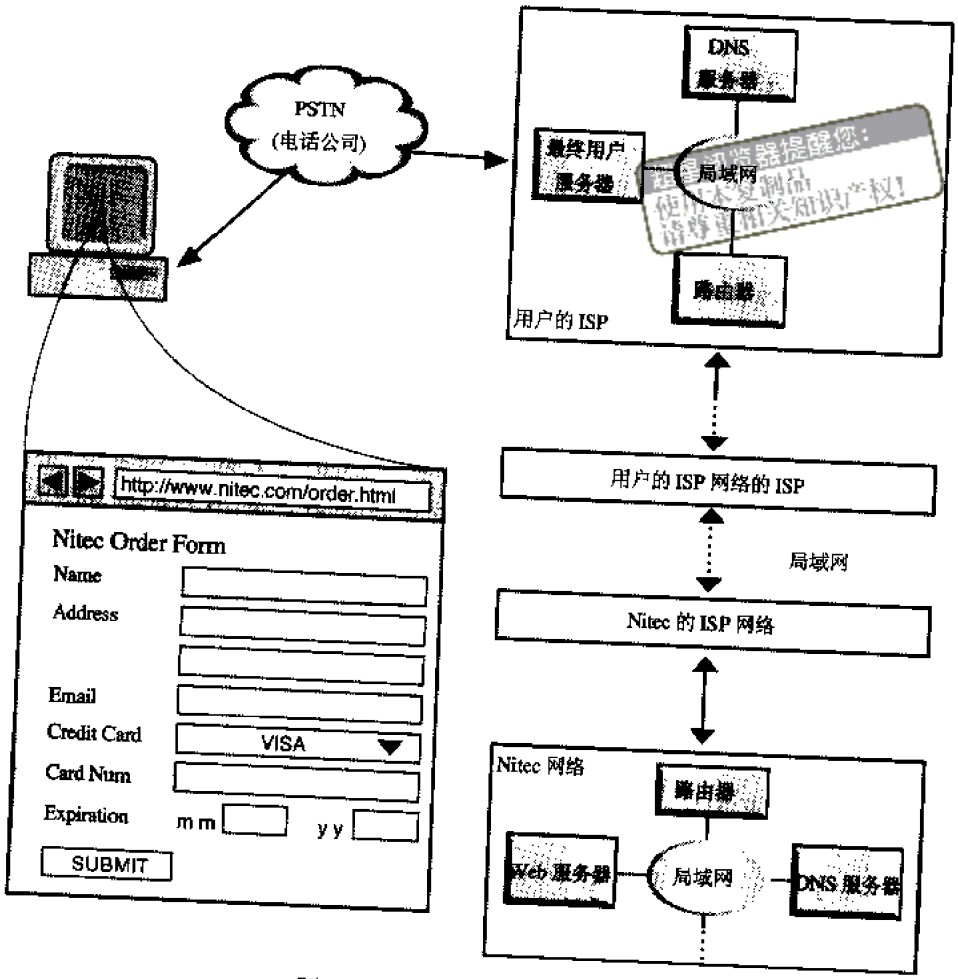


图 15-1 数据在互联网上传递

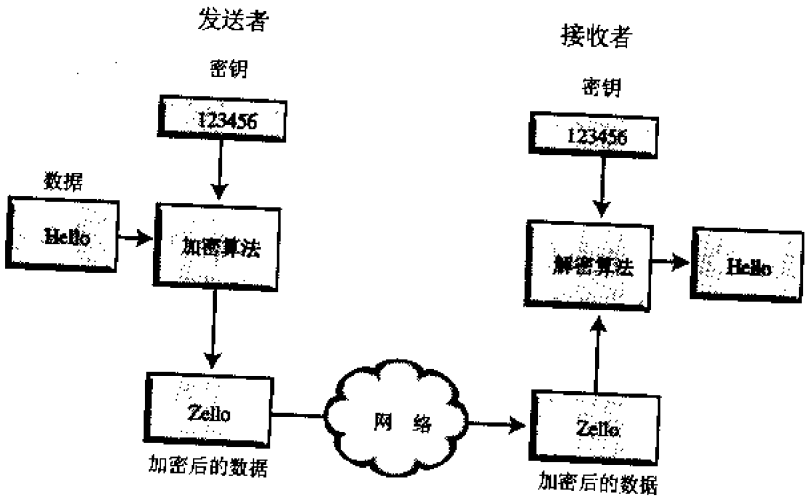


图 15-2 对称加密方案的例子

15.1.2 不对称加密

不对称加密方案中有两把密钥：一把公用密钥、一把专用密钥。额外增加了公用密钥——因此这种方案也称为公用密钥算法。图 15-3 是这种加密方案如何进行工作的例子。

超星阅览器提醒您 使用本复制品请尊重相关知识产权！

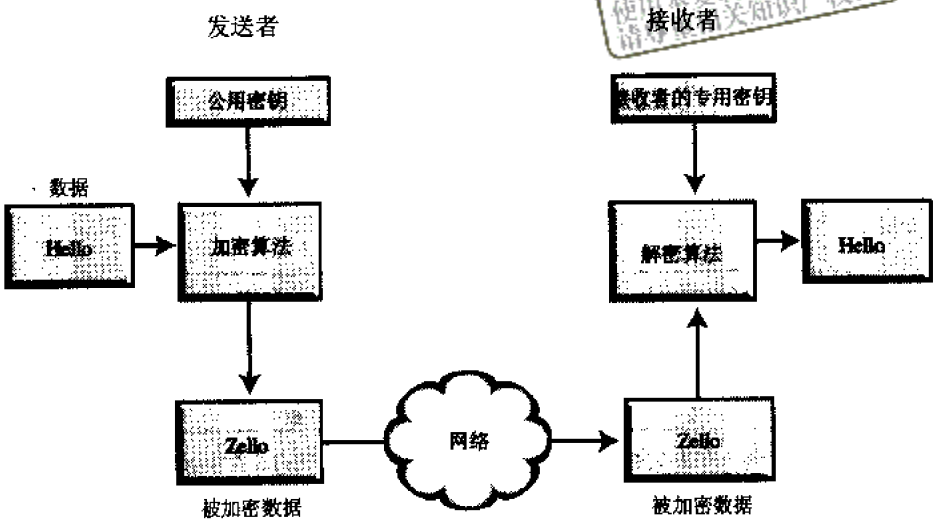


图 15-3 不对称加密方案的例子

如果数据用公用密钥加密，那么只有使用专用密钥才能解密；反之亦然。与对称加密不同的是，发送者不需要知道接收者用于解密的专用密钥。公用密钥是公开的，因此任何想进行安全数据传输的人都可以使用。专用密钥不公开，它是保密的。

同时使用对称加密和不对称加密，Netscape 开发了开放的非专有的协议。它称为安全 Socket 层（SSL），它对基于 TCP/IP 的通信提供数据加密、服务器认证、数据完整性检查、客户认证。

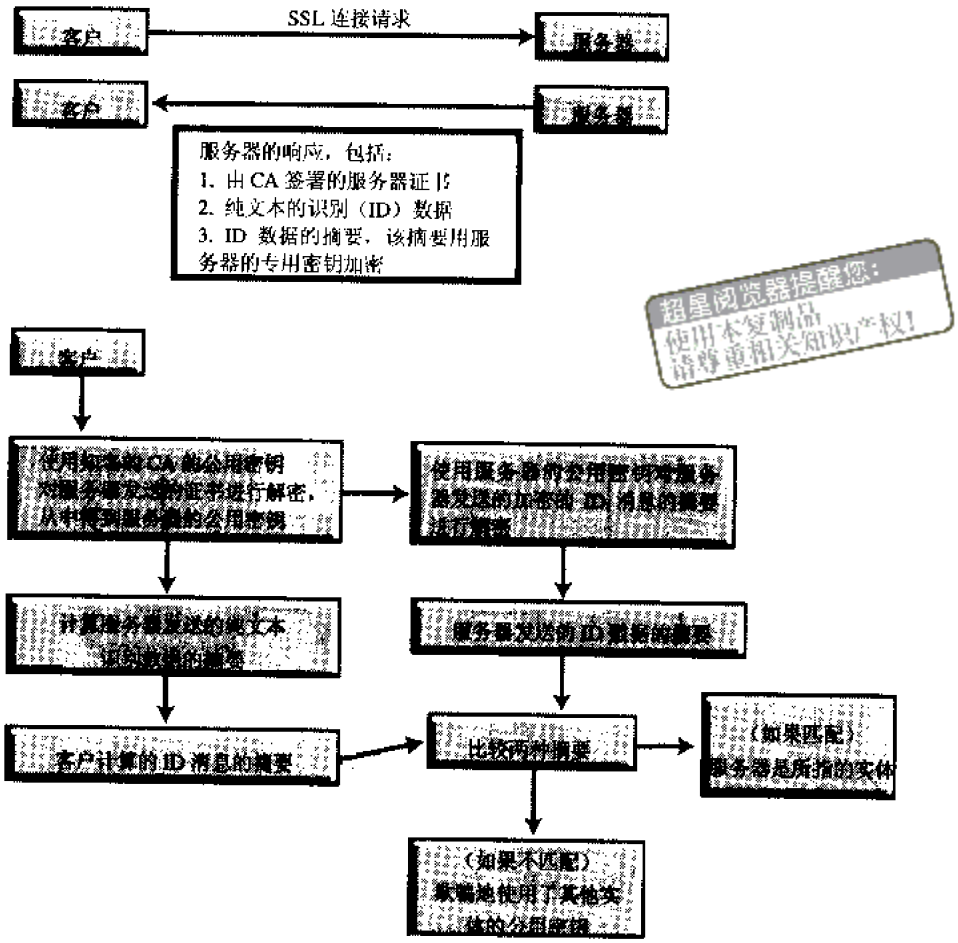
15.2 理解 SSL

在基于 SSL 的传输中（如图 15-4 所示），服务器要向客户系统发送一份证书（certificate）。证书是一些标准数据，它包括：

- 证书发行者的名字
- 发行证书的实体
- 实体的公用密钥
- 其他相关的信息

证书通常由知名的颁发机构发行，这个颁发机构被称为证书颁发（或授权）机构（CA）。证书用 CA 的专用密钥加密。客户对使用 CA 专用密钥加密的证书进行解密。

因为在证书中包含服务器的公用密钥，因此客户可以对服务器加密的任何数据进行解密。到此为止，服务器发送的一些数据用来识别它自己是证书中提及的实体。接着，服务



超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

图 15-4 SSL 的事务处理

器利用前面用于识别它自己的数据产生摘要消息，并用服务器的专用密钥对摘要消息加密。客户现在已拥有已知的 CA 的证书，它说明服务器的公用密钥是什么，来自服务器的识别消息以及识别消息的加密摘要消息。

客户可以使用服务器的公用密钥对摘要消息进行解密。接着，客户可以建立识别消息的摘要，并与服务器发送的摘要相比较。如果匹配，则意味着服务器就是声称的实体。为什么要这样做呢？服务器首先发送由已知的 CA 签署的证书，因此客户知道这个公用密钥属于谁。但是客户需要证明发送证书的服务器就是证书中声称的服务器。因此服务器需要发送简单的识别消息和用它的专用密钥加密的识别消息的摘要。如果服务器没有正确的专用密钥，它不可能产生客户从识别消息中计算出的摘要消息。

这看起来比较复杂，但问题还没有结束。客户现在可以向服务器发送一个对称的加密密钥，并用服务器的公用密钥进行加密。服务器可以用这个新密钥加密数据并把它传给客户。为什么这么做呢？因为使用对称加密比非对称加密快得多。可以这样认为：非对称加密（专用/公用密钥）常用于客户向服务器安全传输随机产生的对称密钥，这个对称密钥随后用于提供一条快速、安全的传输通道。

如果客户和服务器之间有冒名顶替者截获被传输的数据，这将会有什么危害呢？因为它不知道客户、服务器所使用的秘密对称密钥，所以它不能知道数据的内容。但它可以在数据包中插入它自己的数据作为垃圾。

为了避免这种情况发生，SSL 协议允许使用消息认证代码（MAC）。MAC 仅仅是使用对称密钥计算的一些数据，它将被传送。因为冒名顶替者不知道对称密钥，所以它不能计算 MAC 的正确值。例如，一种知名的加密摘要算法叫 MD5，它是由 RSA Data Security 公司开发的。它可以为每个进行传送的数据包产生 128 位的 MAC 值。成功猜出正确的 MAC 需要的巨大的计算能力和时间，因而猜出的可能性几乎是不存在的。

SSL 使得在互联网上进行安全贸易成为可能。但不幸的是，由于法律的限制，Apache 服务器上还不能提供 SSL。当前 SSL 所使用的算法是 RSA 的专利。为了公司的利益，美国在使用公司专利方面有严格的限制。因此，如果 Apache 包含的代码使用了 RSA 提供 SSL 支持，它就不是自由提供的 Web 服务器软件了。另一个原因是美国政府严格限制强加密软件的出口。政府只允许 40 位的加密方案出口到世界上其他国家。但 40 位加密不能提供足够的安全，它已能被破译。同时 Apache 是由国际开发组织开发的，在 Apache 中由于使用强加密来支持 SSL 而引起的各种法律和官僚主义问题在这个开发项目中是不受欢迎的。由于这些问题的存在，所以决定只要这些问题继续存在，Apache 就不直接对 SSL 提供支持。

但不要因此感到泄气，还有其他可行的方案：包括自由提供的和商业的方案。如果想把 SSL 用于非商业的目的，可以用 Apache-SSL。如果出于商业的需要，可以使用商业 Apache 的衍生产品——由 C2Net Technology 公司生产的 Stronghold。在下面的章节中，将详细讨论这两种方案。

15.3 Apache-SSL

Apache-SSL 是一个安全的、非商业的 Apache Web 服务器软件，它基于 Apache 和自由实现的称为 SSLeay 的 SSL 库。这个库支持 SSL 版本 2 和 SSL 版本 3。在进一步利用 Apache-SSL 前，确保了解下面的法律问题。

如果计划在美国以外使用 Apache-SSL，根据 SSLeay 文档，这应该没有法律限制。如果使用了 RSAREF 提供的 RSA 算法，可以把 Apache-SSL 用于非商业的目的。如果用于商业目的，则需要得到 RSA 的许可，因为 RSA 在美国具有专利。同时，他们可能限制在欧洲使用 IDEA 算法，限制在美国使用 RC4。

如果可以（也就是合法的）并愿意试一试 Apache-SSL，第一步是在你的系统中建立 SSLeay。可以在下列站点得到 SSLeay。

```
ftp://ftp.psy.uq.oz.au/pub/Crypto/SSL
```

15.3.1 建立 SSLeay

首先，从前面的站点得到 SSLeay 的压缩 tar 文件。建立 SSLeay 的最容易的方法是运

行称为 Configure 的 Perl 脚本，在它的发行版中提供这个脚本。当运行脚本时需要指定你的操作系统名。如果不带任何参数运行这个脚本，将看到它所支持的操作系统名称列表。

例如，如果想为基于 ELF 的 Linux 系统（如 RedHat 5.0）创建相应的 makfile 配置，可运行如下的脚本：

```
perl  Configure linux-elf
```

如果 Configure 脚本运行成功，可以运行如下命令：

```
make  clean
make
make  rehash
make  test
make  install
```

超星阅览器提醒您：
此为电子复制品
请尊重相关知识产权！

第一个 make 命令清除前面的编译过程中留下的目标文件，第二个 make 命令是编译库，第三个 make 命令是恢复演示证书，第四个 make 命令进行测试。

Note

如果在 Windows 平台上，或者安装出现有问题，可以从如下 Web 站点得到更多的关于如何建立 SSLeay 的信息：

www.cryptsoft.com/ssleay/doc/Building.html

如果成功测试了 SSLeay，就可以运行如下命令安装软件：

```
make install
```

它的可执行文件称为 ssleay，放在 apps 子目录下。可以用下面命令查看版本信息：

```
ssleay  version
```

为你的系统建立 SSLeay 支持后，就可以继续创建 Apache-SSL 了。

15.3.2 建立 Apache-SSL

访问 www.apache-ssl.org/站点，并在其中找到附近的 FTP 站点，就可以得到 Apache-SSL 文件。下面是其中的一个 FTP 的站点：

```
ftp://ftp.ox.ac.uk/pub/crypto/ssl/
```

如果已经得到了 Apache-SSL 的压缩 tar 文件，把这个文件拷到当前 Apache 目录下（即 Apache 的顶级目录）并把它解压缩。现在可以用下面的命令修补原始的 Apache 文件：

```
patch  <  SSLpatch
```

改变目录到 Apache 的 src 子目录下并编辑配置文件。要确保 SSL_BASE 的值设置为解压缩 SSLeay 源文件的目录。如果 SSL_BASE 指向正确的位置，就可以像平常那样运行 Apache 配置脚本 Configure，最后得到 Apache-SSL 的称为 httpsd 的可执行文件。如果想检验该可执行文件的版本，可以试用下面的命令：

```
httpsd  -v
```

除了 Apache 版本号，还可以看到 Ben-SSL 的版本号。现在可以为你的安全 Apache 服务器创建临时证书了。

15.3.3 创建临时证书

证书的建立过程需要一个随机数。为了向这一过程提供随机数据，可以使用任意文本数据建立一个文件。也可以从任何位置拷贝已存在的文本文件。然后，设置称为 RANDFILE 的环境变量使它指向你想使用的文件。例如：

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

```
setenv RANDFILE  /tmp/random.txt
```

这个命令把 RANDFILE 环境变量指向/tmp/random.txt 文件。应该参考关于如何设置环境变量的 shell 文档。前面的命令适用于 tcsh shell。一旦设置了这个变量，可以运行：

```
make  certificate
```

它将提示输入一些一般信息，例如，你的公司名、你的名字和 E-mail 地址等。

当用 make certificate 创建自定义证书时，需要在 Common Name（公共名）字段输入一些内容。应该输入正在运行服务器的主机名作为公共名，而不是用自己的名字，就像提示信息中的例子一样。

证书存储在 SSLconf/conf/httpsd.pem 文件。

15.3.4 配置 Apache-SSL 的命令

现在差不多可以为 SSL 配置 Apache 了。但是，还需要了解 Apache-SSL 提供的命令，将使用它们对 Apache 进行配置。

15.3.4.1 SSLDisable

语　　法：SSLDisable

使用范围：服务器配置，虚拟主机

这个命令禁止使用 SSL。它不需要任何参数。

15.3.4.2 SSLCertificateFile

语　　法：SSLCertificateFile certificate_filename

缺 省 值：无

使用范围：服务器配置，虚拟主机

这个命令为 Web 站点主机指定证书的文件名。注意，需要为每个 Web 站点指定单独的证书。因此，如果计划为虚拟主机提供 SSL 连接，那么这个虚拟主机也需要一个单独的证书。

15.3.4.3 SSLCertificateKeyFile

语　　法：SSLCertificateKeyFile certificate_key_filename

缺 省 值：无

使用范围：服务器配置，虚拟主机

这个命令定义的是证书专用密钥的文件名。如果不使用这个命令，密钥被认为在 SSLCertificateFile 指定的文件中。

15.3.4.4 SSLCACertificatePath