Red Hat Secure Web Server

Getting Started Guide

Red Hat Software, Inc.
Research Triangle Park, North Carolina

Revision: SecServ-2.0-Print-RHS (9/98)

Red Hat Software, Inc.
4201 Research Commons, Suite 100
79 T. W. Alexander Drive
P. O. Box 13588
Research Triangle Park, NC 27709
(919) 547-0012
`redhat@redhat.com`
`http://www.redhat.com`

Contents

Introduction v

Acknowledgements ix

1 Installing Your Apache Server 1

1.1 OS and Software Versions 2

1.2 Mounting the CD-ROM 3

1.3 Optional Packages 3

1.4 Running the Installer 9

2 Configuring Your Secure Web Server 15

2.1 Apache Configuration 16

2.2 httpd.conf 17

2.3 srm.conf 27

2.4 access.conf 32

2.5 Adding Modules to Your Server 34

2.6 Using Virtual Hosts 36

2.7 Starting and Stopping Your Server 40

2.8 Accessing Your Server 42

3 Securing Your Server 43

3.1 How Server Security Works 44

3.2 Deciding on a Certificate Authority 46

3.3 Proving Your Organization's Identity to a CA 46

3.4 Creating Your Key and Certificate Request 49

3.5 Getting a Test Certificate 54

3.6 Installing and Testing Your Certificate 58

3.7 Buying a Certificate 59

4 Configuring Optional Packages 77

4.1 Configuring Analog 77

4.2 Configuring mod_perl 78

4.3 Configuring mod_php 81

4.4 Configuring Apache-ASP 83

4.5 Configuring Squid 83

4.6 Configuring ht://Dig 86

Index 89

Introduction

The *Red Hat Secure Web Server Getting Started Guide* is intended to get you started running your Red Hat Secure Web Server. It is not meant to be complete and exclusive documentation for any of the programs included with this package. When necessary, this guide will point you to the appropriate places where you can find more in-depth documentation on particular subjects.

This guide will show you how to install the included programs, as well as the basic options for configuring your Apache web server. You will also be walked through the steps necessary to get both test and signed certificates, as well as how to install a certificate to use with your secure web server.

After reading and following the steps in this guide, your secure server will be running using a test certificate. If you've followed our instructions for requesting a certificate from the certificate authority of your choice, you'll be ready for secure e-commerce as soon as your certificate arrives.

New features included in Red Hat Secure Web Server version 2.0 include a new version of Apache as well as a new security module. The most significant new feature in version 1.3 of the Apache web server is its support for Dynamic Shared Objects (DSOs). DSO support makes it easier for users to compile and load other modules into their web server. The new version of Apache also offers other improvements and bug fixes.

Version 2.0 of the Red Hat Secure Web Server uses the mod_ssl security module for security instead of Apache-SSL. mod_ssl is partially based on Apache-SSL, but has improved on its predecessor in several different ways:

- mod_ssl provides complete documentation
- mod_ssl has fixed many different bugs that existed in Apache-SSL

Other new features include: the compilation of all Apache modules, additional optional packages like PHP3 and Apache ASP, and improved documentation.

Changes to this manual include more detail on the following subjects:

- configuration of your secure web server
- configuration of virtual hosts
- optional packages supplied with your secure web server
- Apache and mod_ssl configuration directives
- web server security

This manual no longer includes the mod_php (PHP/FI) functions which were included as Appendix A in version 1.0. If you need to use those functions, a complete list (including descriptions) is available from the PHP website at `http://www.php.net/manual/phpfi2.html#funcs`. If you intend to use PHP3 instead of PHP/FI, information about PHP3 functions can also be found at the PHP website at `http://www.php.net/quickref.php3`.

We Need Feedback!

If you've found a mistake in this manual, or if you've thought of a way to make it better, we'd love to hear from you! Please send mail to:

`docs@redhat.com`

Be sure to mention the manual's identifier:

SecServ-2.0-Print-RHS (9/98)

If you include the manual's identifier, we'll know exactly which version of this manual you have. If you have a suggestion, try to be as specific as possible. If you've found an error, please include the section number and some of the surrounding text so we can find it easily. We may not be able to respond to every message sent to us, but you can be sure that we'll be reading them all.

Acknowledgements

Red Hat Software would like to acknowledge the following contributions to this product:

This product includes software developed by the Apache Group for use in the Apache HTTP server project (http://www.apache.org/).

This product includes mod_ssl software developed by Ralf S. Engelschall (http://www.engelschall.com/sw/mod_ssl/).

This product includes software developed by Ben Laurie for use in the Apache-SSL HTTP server project (http://www.apache-ssl.org/.

The product includes SSLeay cryptographic software written by Eric Young (http://www.ssleay.org/).

1

Installing Your Apache Server

After you have read this chapter and followed the instructions it contains, your web server will be installed and configured. You'll also be taught how to start your web server and run it without security in order to test your installation.

Please Note: In order to install the Red Hat Secure Web Server, you *must* already have obtained and installed the Red Hat Linux operating system on your secure web server's system. Red Hat Linux is not included with the Red Hat Secure Web Server product.

Before you begin the installation process, if you are running any web server, you must stop the server process. If you are running an Apache web server, stop the server process by issuing the appropriate command or commands from the following list:

```
/etc/rc.d/init.d/httpsd stop
/etc/rc.d/init.d/httpd stop
```

(In other words, if your system only has the script `/etc/rc.d/init.d/httpd`, then execute that script with the `stop` parameter. If it has both scripts, execute both, and so on.)

If you have another version of Apache installed and you customized its configuration files, its configuration files will be saved in their directory with an extension of `.rpmsave` during the installation of your new secure web server. If you had another version of Apache installed but you never customized its configuration files, they will be written over during the installation of this product.

If you have the previous version of the Red Hat Secure Web Server installed, you should stop the server process as described above before installing this product as described in this chapter.

1.1 OS and Software Versions

If you are using Red Hat Linux version 3.0.3 or earlier, you should upgrade your system to a more recent version (preferably 5.x) before installing the Red Hat Secure Web Server. If you don't upgrade, the installation won't be accomplished using RPM. If you install this product without using RPM, it will be much more difficult to remove the secure server software at a later time. In addition, the C libraries on your system will also be significantly older than the libraries used to build the server. If you use older C libraries with the secure web server, you may run into unexpected problems.

When you run the installer, you may see a message which warns you that your version of RPM is old and asks if you want to upgrade. You should choose to upgrade. If you do not, the installer will use `cpio` to perform the secure web server install, making the secure web server difficult to uninstall later. Upgrading RPM shouldn't adversely affect your system, so go ahead and do it.

During the installation process, you may also see a message about `glibc` being too old. You should allow the installer to upgrade `glibc`. If you don't, the rest of the installation may not go smoothly.

1.2 Mounting the CD-ROM

To begin the installation process, you must first mount the CD-ROM. Place the secure web server CD in your CD-ROM drive. Then, as root, type the following command to mount the installation CD:

```
mount -t iso9660 /dev/cdrom /mnt/cdrom
```

Please Note: On your system, you or the system administrator may already allow users (instead of only root) to mount the CD-ROM drive. Users have this privilege if the `user` option is included in the `/dev/cdrom` line in the `/etc/fstab` file.

Even if the users can mount the CD-ROM drive, however, the installation program will not work unless they also have the `exec` option set in the `/dev/cdrom` line in `/etc/fstab`. To add the `exec` option, edit the `/dev/cdrom` line in `/etc/fstab`. The original line:

```
/dev/cdrom /mnt/cdrom iso9660 user,noauto,ro 0 0
```

should be changed to:

```
/dev/cdrom /mnt/cdrom iso9660 user,noauto,ro,exec 0 0
```

1.3 Optional Packages

When you run the installer, you will be asked which additional packages you would like to install. We are providing short descriptions of these optional packages, so that you may make an informed decision about whether to install each optional package or not. The following sections provide these brief descriptions and include the post-installation location of each package's documentation and configuration files.

When possible, we've also included a reference to a web page where you should be able to find more information about configuring and managing the program. Remember, however, that these web pages may include information about a more recent version of the particular package, if a new version of the package has become available since the release of this version of the Red Hat Secure Web Server.

Before you start the installation, please review the following packages and decide which ones you want to install with your secure web server.

1.3.1 Analog

Configuration File: `/etc/analog.cfg`

Documentation: `/usr/doc/analog-3.0/`

Description: Analog is a program that analyzes your web server's logfiles. Analog parses your web server's logfiles to provide you with lots of valuable (or at least interesting) statistics and information. For example, Analog can tell you how often web pages on your server are retrieved, from what countries the requests are originating, which web sites include broken links, and more. Analog's reports are normally viewed through its web interface, which you can access using almost any browser.

For more information about configuring Analog after installation, see section 4.1 on page 77. You may also want to try the Analog web page at `http://www.statslab.cam.ac.uk/~sret1/analog/`.

1.3.2 mod_perl

Configuration File: N/A

Documentation: `/usr/doc/mod_perl-1.15/` or use the commands:

```
perldoc mod_perl
perldoc Apache
```

Description: mod_perl is an Apache module that incorporates a Perl interpreter into the Apache web server, so the Apache web server can directly execute Perl code. Installation of the `mod_perl` package links the Perl runtime library into the server and provides an object-oriented Perl interface for the Apache server's C language Application Programming Interface (API). The end result is a quicker CGI script turnaround process, since no external Perl interpreter has to be started.

The most common use of mod_perl is the use of its Apache::Registry module as a speedy replacement for the Common Gateway Interface (CGI). The Apache::Registry module emulates the CGI environment, so programmers can write CGI scripts which will run under either CGI or mod_perl.

You should realize that previously existing CGI scripts may require some improvements. Normally, CGI scripts have a lifetime of one HTTP request. That short lifespan allows programmers to get away with questionable scripting. Since the Apache::Registry module maintains a cache of scripts, it is quicker, but it may be less forgiving of non-standard programming.

For more information on how to set up mod_perl as a replacement for CGI, refer to section 4.2 on page 78. For more general information about mod_perl, try the Apache/Perl Integration Project web page at `http://perl.apache.org/`.

1.3.3 PHP3 and PHP/FI

Configuration File: N/A

Documentation: `/usr/doc/mod_php-3.0.3/` or `/usr/doc/mod_php-2.01/`

Description: PHP is an HTML-embedded scripting language. PHP attempts to make it easy for developers to write dynamically generated web pages. PHP also offers built-in database integration for several commercial and non-commercial database management systems, so writing a database-enabled web page with PHP is fairly simple. The most common use of PHP coding is probably as a replacement for CGI scripts.

The mod_php module enables the Apache web server to understand and process the embedded PHP language in web pages. Please refer to section 4.3 on page 81 for more information on post-installation configuration of mod_php. You should also try the PHP web page at http://www.php.net/ for more information about PHP.

You may install either PHP3 (the mod_php3 package) or PHP/FI (the mod_php package) or both. We are providing PHP/FI for people who run other programs which depend on PHP/FI and will not run with PHP3. If you have never used PHP/FI before, but you would now like to try PHP, you do not need to install the mod_php package; you should install the mod_php3 package.

1.3.4 Apache-ASP

Configuration File: N/A

Documentation: /usr/doc/perl-Apache-ASP-0.02/

Description: Apache ASP is a port of Active Server Pages (ASP) to Apache. Theoretically, Apache ASP allows developers to create ASP-style web applications that embed session management and perl into HTML files.

If you are going to install the ASP package, you'll also need to install the mod_perl package.

1.3.5 Devel

Configuration File: N/A

Documentation: N/A

Description: The devel package (secureweb-devel) contains the Apache include files, header files and the APXS utility. You will need all of these things if you intend to load in any extra modules, other than the modules provided with this product, to your secure web server. Please see section 2.5 on page 34 for more information on loading modules in to your secure web server using Apache's Dynamic Shared Object (DSO) support.

If you do not intend to load in other modules to your secure web server, you do not need to install this package.

1.3.6 Source

Configuration File: N/A

Documentation: N/A

Description: The `source` package (`secureweb-source`) contains the Apache source code for your secure web server. You need to install `source` if you plan on including an extra module that needs the source code in order to compile. See section 2.5 on page 34 for more information about including modules using Apache's DSO support.

Unless you plan on compiling and loading a module which will require the source code, you do not need to install this package. Most modules will not need the source code in order to compile. If you do install this package, however, you will also need to install the `devel` package.

1.3.7 Squid

Configuration File: `/etc/squid.conf`

Documentation: `/usr/doc/squid-1.1.22/`

Description: Squid is a proxy caching server for web clients which supports HTTP, FTP and gopher data objects. Squid keeps meta data, popular objects, and Domain Name Server (DNS) lookups cached in RAM (or on disk if you don't have the memory to spare). Squid supports non-blocking DNS lookups and implements negative caching of failed requests.

Using Squid, you can set web browsers to use your web server as a proxy server. Obviously, this is only useful if you have more than one person using it or you repeatedly visit the same web pages. Squid will cache requests so that if you access a site more than once, the subsequent retrievals will be much faster. The second and subsequent retrievals will be retrieved from the proxy server's memory instead of from the actual website.

When you install the `squid` package, you can choose whether to install the memory caching version (if you have memory to spare) or the disk caching version (described below). See section 4.5 on page 83 for more information on configuring Squid after installation. You may also want to try the Squid web page at `http://squid.nlanr.net/` for more information.

1.3.8 Squid-novm

Configuration File: `/etc/squid.conf`

Documentation: `/usr/doc/squid-novm-1.1.22/`

Description: Squid-novm is the same as the Squid package except that it uses your disk drive instead of your RAM to hold the cache.

1.3.9 ht://Dig

Configuration File: `/etc/htdig.conf`

Documentation: `/usr/doc/htdig-3.0.8b2/`

Description: ht://Dig is a web indexing and search engine intended to be used by small domains or intranets. ht://Dig isn't meant to be a "real" Internet search engine like AltaVista, Excite, etc. Instead, ht://Dig is meant to provide searching capabilities for a single company or campus website or even for a subsection of a large website.

Please Note: ht://Dig does not currently have the ability to connect to a secure web server. If you want to use ht://Dig with your Red Hat Secure Web Server, you will need to leave your server's configuration at the default configuration, which enables both secure and non-secure operations. Please see section 2.6 on page 36 for information on how the default configuration of your secure web server runs secure and non-secure servers on your machine using virtual hosts.

See section 4.6 on page 86 for more information on how to configure ht://Dig after installation. You may also want to look for more information on the ht://Dig web page at `http://www.htdig.org`.

1.3.10 Netscape Navigator

Configuration File: N/A

Documentation: `http://help.netscape.com/`

Description: Netscape Navigator is a popular web browser. Extensive help is available using Navigator's 'Help' menu or from Netscape's Technical Support web site at `http://help.netscape.com`. This particular version of Navigator is version 4.06.

1.3.11 Netscape Communicator

Configuration File: N/A

Documentation: `http://help.netscape.com/`

Description: Netscape Communicator is a suite of products that includes the Navigator web browser as well as an email client, a news reader, and a web page editor. Extensive help is available via any of the individual program's 'Help' menus or from Netscape's Technical Support web site at `http://help.netscape.com`.

1.4 Running the Installer

If you've decided which optional packages you are going to install with the secure web server, you're ready to start the installation process.

You should be using the console or a color xterm (xterm-color). Change the working directory to your CD-ROM's mount point:

```
cd /mnt/cdrom
```

Type in the next command to run the installer:

```
./install-webserver
```

You'll see a window like the one shown in figure 1.1, thanking you for purchasing Red Hat Secure Web Server 2.0. Press the Enter key to choose **Ok** and continue with the installation.

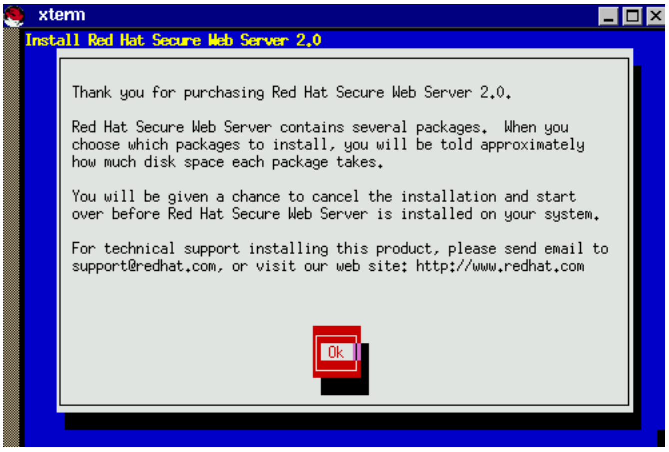

Figure 1.1: Starting the Install

Follow the applicable directions outlined next:

1. **Upgrade Required Software**
   If you have older versions of RPM or glibc, you will be notified and asked to upgrade. You should choose to upgrade.

2. **Selecting Packages**
   Select the optional packages that you wish to install from the list provided. See figure 1.2 on the next page. Use the ↑ and ↓ keys to move the cursor up and down the list. Note that you can ↓ down to see more available packages. Press the spacebar to select or deselect each package. When you've selected all of the optional packages that you would like to install, press the tab key to move to the **Ok** button and press Enter.

   (a) **If you selected squid**
   If you chose to install the `squid` package, you will see a dialog box which asks whether you want to place Squid's cache in

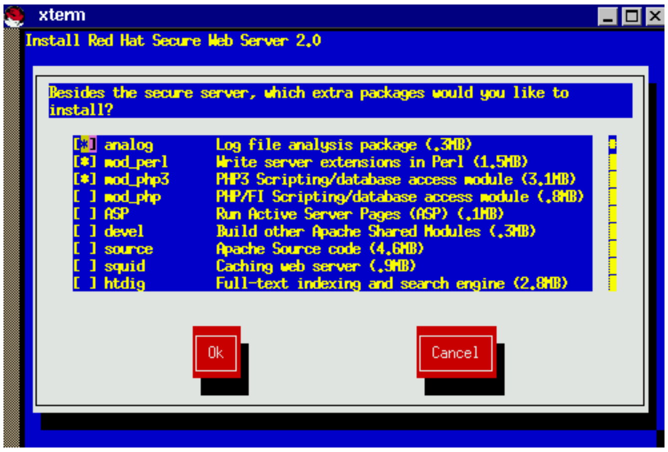

Figure 1.2: Optional Packages to Install

memory or on disk. If your server is equipped with plenty of memory (i.e., 64MB or more), you should choose **squid** so that the cache will be placed in memory (and will be faster). If you have less than 64MB of memory, choose **squid-novm** and the cache will be created on disk.

Click the space bar to select either **squid** or **squid-novm**. Tab to the **Ok** button and press Enter.

(b) **If you selected analog**

If you chose to install analog, the next screen you'll see will contain the dialog box as shown in figure 1.3 on the following page. This dialog box asks if you want to install Analog's web-based interface. If you've chosen to install Analog, you will probably want to use its web-based interface because it is the easiest way to manage and use Analog.

Use the tab button to choose **Yes** or **No** and then press Enter.

3. **Continue with the Install**

At this point, the installer will display a dialog box (see figure 1.4 on page 13) which tells you the total amount of disk space required for the packages you selected. If the installation will take up too much

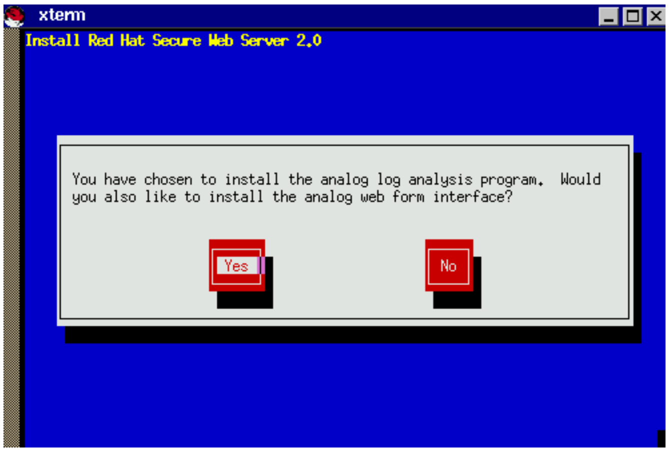

Figure 1.3: Analog Package Options

space on your hard disk, select **No**, and re-run the installation selecting fewer optional packages. If the installation size is acceptable, select **Yes**, and press Enter.

4. **Installing Packages**

   If you selected **Yes**, the installer will display progress bars as RPM inspects your system and then as it installs each package. One of the progress bars is shown in figure 1.5 on the facing page.

   When the install is complete, the installer will display a dialog box as shown in figure 1.6 on page 14. Press Enter and you will be returned to a shell prompt. The next step is to configure your secure web server.

   Please Note: At the very end of the installation, you may see an error message if (1) you are running the Red Hat Linux operating system version 4.2 and (2) you had the *original* Apache web server version 1.1.3 (i.e., with *no* updates) installed before you began the secure web server installation. The error message will warn you that `/etc/httpd` cannot be removed. Press Enter to accept the **Ok** option and ignore this error message. Your secure web server has been successfully installed.

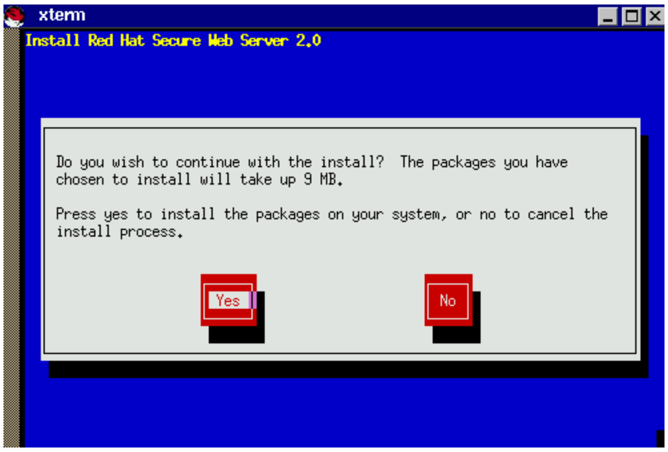

Figure 1.4: Continue with Installation

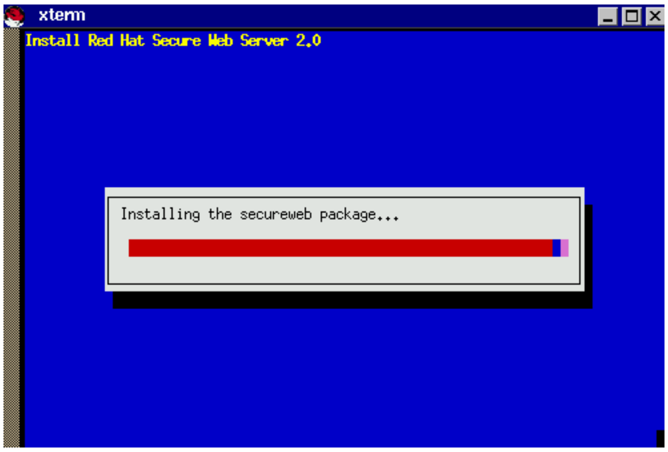

Figure 1.5: Installation Status Bar

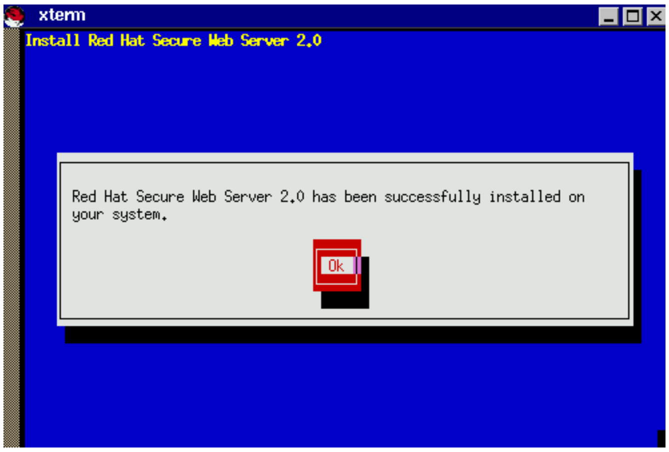

Figure 1.6: Installation Complete

2

Configuring Your Secure Web Server

You can't start your secure web server right now, because you haven't created your key or obtained a digital certificate yet. By default, your secure web server needs those security files to work. Chapter 3 on page 43 covers how to create your key and certificate request and how to install your digital certificate.

Before you start with the security considerations, however, you should become familiar with some of the configuration options for your secure web server. You shouldn't need to change any of the default configuration options, but you should know what some of the options are, and know where to find them.

Once you get your server running (after you create and install a digital certificate) you can access the full Apache server documentation at `http://www.yourdomain.com/manual/` or you can use the Apache documentation available on the web at `http://www.apache.org`. The Apache server documentation contains a full list and complete descriptions of all of Apache's configuration options. For your convenience, short descriptions of the configuration directives used in your secure web server are provided in this manual.

When you are looking through your web server's configuration files, be aware that your default configuration includes both a non-secure web server and a secure web server. The secure web server runs as a virtual host, which is configured in the `httpd.conf` configuration file. For more information about virtual hosts, see section 2.6 on page 36.

Please Note: We do not include FrontPage extensions, which would give the Red Hat Secure Web Server the capability of working with Microsoft FrontPage. Including FrontPage support is a problem for two reasons. First, the Microsoft license prohibits the inclusion of the extensions in a third party product. Secondly, the patch which would enable Apache to include FrontPage extensions is a very serious security risk. Much of the workaround involves granting root access, which is something that you want to avoid for security's sake. This type of risk should be unacceptable to anyone who is even somewhat concerned with the security of their secure web server and their website.

2.1 Apache Configuration

`/etc/httpd/conf/` is the default location for the Apache web server's configuration files: `httpd.conf`, `srm.conf` and `access.conf`. All three of these files are well-commented and somewhat self-explanatory. You will probably not need to change the information they contain, but you should be familiar with their most important configuration options.

If you do need to configure your secure web server, you simply edit one of the three files, then either reload, or stop and start your secure web server. How to reload, stop and start your server is covered in section 2.7 on page 40.

Before you edit one of the configuration files, it is a good idea to first copy the original file to something like `httpd.confold` (or to any name you want). Then, if you make a mistake while you're editing the configuration file, you'll have a backup to start over with.

If you do make a mistake, and your secure web server doesn't work correctly, the first place to look is in the configuration file you just edited. Make sure that you didn't make a typo. The second place to look is in your secure web server's error log, `/var/log/httpd/error_log-ssl` or in your non-secure web server's error log, `/etc/httpd/logs/error_log`. The error log may not be easy to interpret if you're new to this kind of thing. If you've just experienced a problem, however, the last entries in the error log should help provide some clues about what happened.

The next sections provide short descriptions of the directives which are included in your web server's configuration files. These descriptions are by no means exhaustive. If you need more information, please refer to the Apache documentation provided in HTML format at `http://www.yourdomain.com/manual/` or to the Apache group documentation at `http://www.apache.org/docs/`. For more information about mod_ssl directives, refer to the documentation included in HTML format as `http://www.yourdomain.com/manual/mod/mod_ssl.html`.

2.2 httpd.conf

`httpd.conf` is the primary configuration file for your web server. The directives in `httpd.conf` control exactly how your server operates and how and where logs will be kept.

First, we'll describe the more important directives, which you should be most familiar with, in the order that you'll find them in the `httpd.conf` file.

After the more commonly encountered directives, we'll describe the rest of the directives in `httpd.conf`, also in the order in which they are found in the file. You will not need to change any of these directives in order to use your secure web server in most configurations.

2.2.1 Important Directives in httpd.conf

LoadModule `LoadModule` is used to load in Dynamic Shared Object (DSO) modules. More information on the secure web server's DSO support, including exactly how to use the `LoadModule` directive, can be found in section 2.5 on page 34.

AddModule `AddModule` is the directive used by the secure web server to create a complete list of all available modules. You will use the `AddModule` directive if you add your own module in as a DSO. For more information on how `AddModule` is used for DSO support, see section 2.5 on page 34.

HostnameLookups `HostnameLookups` can be set to `on` or `off`. If you allow `HostnameLookups` (by setting it to `on`), your server will automatically resolve the IP address for each connection which requests a document from your web server. Resolving the IP address means that your server will make one or more connections to the Domain Name System (DNS) in order to discover the hostname that corresponds to a particular IP address.

Generally, you should leave `HostnameLookups` set to the default setting of `off`, because they add a load to your server and may slow it down. If your server is busy, the effects of `HostnameLookups` may be especially noticeable.

`HostnameLookups` are also an issue for the Internet as a whole. All of the individual connections made to look up each hostname add up to significant traffic on the Internet. Therefore, for your own web server's benefit, as well as for the good of the Internet as a whole, you should leave `HostnameLookups` set to `off`.

User The `User` directive sets the userid used by the server to answer requests. `User`'s setting determines the server's access. Any files inaccessible to this user will not be accessible to your web site's visitors. For security reasons, the `User` should only have privileges so that it can access files which are supposed to be visible to the outside world.

The `User` is also the owner of any CGI processes spawned by the server. The `User` should not be allowed to execute any code which is not intended to be in response to httpd requests. The default for `User` is `nobody`, an unprivileged user.

Please Note: Unless you know exactly what you're doing, don't set the `User` to `root`, which will create some big security holes for your secure web server.

The parent `httpd` process first runs as `root` during normal operations, but is then immediately handed off to the `nobody` user. The server has to start as `root` because it needs to bind to a port below 1024 (the default port for secure web communications is port 443, the default port for non-secure web communications is port 80). Ports below 1024 are reserved for system use, so they can't be used by anyone but `root`. Once the server has attached itself to its port, it hands the process off to the `User` before it accepts any connection requests.

Group The `Group` setting is similar to the `User` setting. The `Group` sets the group under which the server will answer requests. The default `Group` is also `nobody`.

ServerAdmin `ServerAdmin` should be the email address of the secure web server's administrator. This email address will show up in error messages on server-generated web pages, so users can report a problem by sending email to the server administrator. `ServerAdmin` it set by default to `root@localhost`, but you may want to change it to your webmaster's email address.

ErrorLog `ErrorLog` names the file where server errors are logged. The error log file for your non-secure web server is located in `/etc/httpd/logs` and is named `error_log`. The `error_log` file for your secure web server is located in the `/var/log/httpd` directory and is named `error_log-ssl`.

Error logs are a good place to look if your web server ever generates any errors or fails and you aren't sure what happened.

CustomLog `CustomLog` is a way of defining a logfile format. In your secure web server's default configuration, `CustomLog` defines the log file where accesses to your non-secure web server are recorded: `/etc/httpd/logs/access_log`. You'll need to know the location of this file if you want to generate any access-based server performance statistics for your non-secure web server. Analog, which you may install along with your secure web server, is a program which parses the access log to generate statistics.

Note that the default `TransferLog` (or access log) for your secure web server is `/var/log/httpd/access_log-ssl`.

2.2.2 Other Directives in httpd.conf

You will not need to change any of the following directives, unless you do some fairly elaborate re-configuration of your secure web server.

ClearModuleList The `ClearModuleList` directive is located immediately before the long list of `AddModule` directives. `ClearModuleList` erases the server's built-in list of active modules. Then the list of `AddModule` directives re-creates the list, immediately after `ClearModuleList`.

ServerType Your `ServerType` can be either `inetd` or `standalone`. By default, your secure web server is set to `ServerType standalone`. `standalone` means that the server is started once and that server handles all of the connections. `ServerType inetd` means that for every http connection, a new instance of the server is started. That server handles the connection and exits when the connection is ended. As you can probably imagine, using `inetd` is very inefficient. Another problem is that `inetd` may not work correctly, according to the Apache group. For those two reasons, you'll want to leave your secure web server's `ServerType` set to `standalone`.

Port Normally, `Port` defines the port that your server is listening to. Your secure web server, however, is listening to more than one port by default, since the `Listen` directive is also being used. When `Listen` directives are in effect, your server listens at all of those ports. See the description of the `Listen` directive for more information about `Listen`.

The `Port` command is also used to specify the port number used to construct a canonical name for your secure web server. See the `UseCanonicalName` directive for more information about your server's canonical name.

Listen The `Listen` command name the ports on which your secure web server will accept incoming requests. Your secure web server is set to listen to port 80 for non-secure web communications and port 443 for secure web communications.

Listen can also be used to specify particular IP addresses over which the server will accept connections.

ServerRoot The ServerRoot is the directory which will contain subdirectories for the server's files. Both your secure and non-secure web servers are set to use a ServerRoot of /etc/httpd.

BindAddress BindAddress is a way of specifying which IP addresses your server will listen to. You can (and probably should) use the Listen directive instead if you need this functionality. BindAddress is not used by your web server; by default it is commented out in httpd.conf.

LogLevel LogLevel sets how verbose the error messages in the error logs will be. LogLevel can be set (from least verbose to most verbose) to emerg, alert, crit, error, warn, notice, info or debug. Your secure web server's LogLevel is set to warn.

LogFormat The LogFormat directives in your httpd.conf file set up a format for the messages in your access log that will make your access log more readable.

PidFile PidFile names the file in which the server records its process id (pid). Your secure web server is set to record its pid in /var/run/httpsd.pid.

ScoreBoardFile The ScoreBoardFile stores internal server process information, which is used for communication between the parent server process and its children processes. Your secure web server's ScoreBoardFile is set to /var/run/httpsd.scoreboard.

LockFile LockFile sets the path to the lockfile used when the Apache server is compiled with either USE_FCNTL_SERIALIZED_ACCEPT or USE_FLOCK_SERIALIZED_ACCEPT. LockFile should normally be left at its default value; it is commented out by default since your secure web server in its default configuration will not need it.

ServerName You can use ServerName to set a host name for your server which might be different from your host's real name. For example, you might want to use `www.yourserver.com` when your server's real name is actually `blah.yourserver.com`. Note that the `ServerName` has to be a valid DNS name that you have the right to use (i.e., you can't just make something up).

If you do specify a `ServerName`, be sure its IP address and server name pair are included in your `/etc/hosts` file.

UseCanonicalName `UseCanonicalName` is set by default to `on`. `UseCanonicalName` allows the server to construct an URL that references itself, using `ServerName` and `Port`.

CacheNegotiatedDocs By default, your secure web server asks proxy servers not to cache any documents which were negotiated on the basis of content (i.e., they may change with time or based on the input from the requester). If you uncomment `CacheNegotiatedDocs`, you are disabling that function and proxy servers will be allowed to cache the documents from then on.

Timeout `Timeout` defines, in seconds, the amount of time that your server will wait for receipts and transmissions during communications. Specifically, `Timeout` defines how long your server will wait to receive a GET request, how long it will wait to receive TCP packets on a POST or PUT request and how long it will wait between ACKs responding to TCP packets. `Timeout` is set to 300, which is appropriate for most situations.

KeepAlive `KeepAlive` allows persistent connections (i.e., more than one request per connection), or does not allow persistent connections. `KeepAlive` can be used to prevent any one client from consuming too much of the server's resources. By default, `KeepAlive` is set to `on`, which means that your server allows persistent connections. You could set it to `off`, which would disable persistent connections. See the `MaxKeepAliveRequests` directive for a related way of limiting requests per connection.

MaxKeepAliveRequests This directive sets the maximum number of requests allowed per persistent connection. Apache recommends a high setting, which will improve your server's performance. `MaxKeepAliveRequests` is set to `100` by default, which should be appropriate for most situations.

KeepAliveTimeout `KeepAliveTimeout` sets the number of seconds your server will wait for a subsequent request, after a request has been served, before it closes the connection. Once a request has been received, the `Timeout` directive applies instead.

MinSpareServers and MaxSpareServers The Apache web server dynamically adapts to the perceived load by maintaining an appropriate number of spare server processes based on the traffic. The server checks the number of servers waiting for a request and kills some if there are more than `MaxSpareServers` or creates some if the number of servers is less than `MinSpareServers`. Your server's default `MinSpareServers` is `8`; your server's default `MaxSpareServers` is `20`. These default settings should be appropriate in almost all situations. You should not increase the `MinSpareServers` to a very large number, since that will create a heavy processing load on your server even when traffic is light.

StartServers `StartServers` sets how many server processes are created upon startup. Since your web server dynamically kills and creates server processes based on traffic load, you won't ever need to change this parameter. Your web server is set to start ten server processes at startup.

MaxClients `MaxClients` sets a limit on the total number of server processes (i.e., simultaneously connected clients) that can run at one time. You want to keep `MaxClients` at a high number (your server's default is set to 150), because no one else will be allowed to connect once that number of simultaneously connected clients is reached. You can't set `MaxClients` to higher than 256 without recompiling Apache. The main reason for having `MaxClients` is so that a runaway web server doesn't crash your operating system.

MaxRequestsPerChild `MaxRequestsPerChild` sets the total number of requests each child server process serves before the child dies. The main reason for setting `MaxRequestsPerChild` is to avoid long-lived process induced memory leaks. The default `MaxRequestsPerChild` for your server is `100`.

ProxyRequests If you uncomment `ProxyRequests` (which is set to `on` but commented out), your Apache server will also function as a proxy server. If you are enabling the proxy server, you will want to uncomment the `Cache` directives to enable proxy caching for your proxy server. Apache proxy serving is enabled by the `proxy_module` which was compiled in and loaded with your web server by default.

Cache directives A number of cache directives are commented out in your `httpd.conf` file. If you are using the proxy server functionality, you should uncomment these directives to enable the proxy cache. The default settings for your cache directives should be appropriate for most configurations.

CacheRoot `CacheRoot` sets the name of the directory which will contain cached files. The default `CacheRoot` set is `/var/cache/httpd`.

CacheSize `CacheSize` sets how much space the cache can use, in KB. The default `CacheSize` is `5` KB.

CacheGcInterval `CacheGcInterval` sets a number of hours. After that number of hours, files in the cache will be deleted if the cache is using more space than allowed by `CacheSize`. The default for `CacheGcInterval` is four hours.

CacheMaxExpire Cached HTML documents will be retained (without a reload from the originating web server) in the cache for a maximum number of hours set by `CacheMaxExpire`. The default is `24` hours.

CacheLastModifiedFactor The `CacheLastModifiedFactor` affects the creation of an expiry date for a document which did not come from its originating server with its own expiry set. The default `CacheLastModifiedFactor` is set to `0.1`, meaning that the expiry date for such documents equals one-tenth of the amount of time since the document was last modified.

CacheDefaultExpire `CacheDefaultExpire` is the expiry time in hours for a document that was received using a protocol that doesn't support expiry times. The default is set to one hour.

NoCache Any document that is retrieved from a host and/or domain that matches one set in `NoCache` will not be cached.

VirtualHost `<VirtualHost>` and `/VirtualHost>` tags surround any configuration directives which are intended to apply to a virtual host (a separate server which runs alongside your default Apache web server). Most configuration directives can be used within virtual host tags, in which case they only apply to that particular virtual host. Please see section 2.6 on page 36 for more information about virtual hosts.

IfModule `<IfModule>` and `</IfModule>` tags surround directives that are conditional. The directives contained within the `IfModule` tags are processed under one of two conditions. The directives are processed if the module contained within the starting `<IfModule>` tag is compiled in to the Apache server. Or, if a "!" (an exclamation point) is included before the module name, the directives are processed only if the module in the starting `<IfModule>` tag is *not* compiled in.

By default, `<IfModule mod_ssl.c>` and an ending tag surround the virtual host tags for your secure web server. Therefore, the virtual host tags containing your secure web server, and all of the directives contained within them, are only processed if the `mod_ssl.c` module was compiled in to your Apache server. As shown in the `AddModule` section of `httpd.conf`, that particular module was compiled in during installation.

2.2.3 SSL Directives in httpd.conf

The SSL directives in your server's `httpd.conf` file are included to enable secure web communications. Short descriptions of the SSL directives used in your `httpd.conf` configuration file are provided next. For more information on SSL directives, please point your browser to `http://www.yourdomain.com/manual/mod/mod_ssl.html`. More information is also available from the mod_ssl website at `http://www.engelschall.com/sw/mod_ssl/`. For more information about securing your web server, see chapter 3 on page 43.

Please Note: Don't modify your SSL directives unless you're absolutely sure about what you're doing. For the vast majority of secure web servers, the SSL directives are configured appropriately as installed.

SSLDisable `SSLDisable` disables the SSL protocol engine. The

`SSLDisable` directive is used to disable SSL for your non-secure web server.

SSLEnable `SSLEnable` enables the SSL protocol engine. The `SSLEnable` directive is used to enable SSL within the virtual host tags for your secure web server.

SSLRequireSSL `SSLRequireSSL` denies access to all requests unless SSL is in use. This is a good security safeguard, since its use provides a backup in case of any configuration errors that could leave normally secure documents unprotected.

SSLCertificateFile The `SSLCertificateFile` states the path to your certificate. After your certificate is created, it will be saved in `/etc/httpd/conf/httpsd.crt`, your secure web server's `SSLCertificateFile`.

SSLCertificateKeyFile The `SSLCertificateKeyFile` states the path to your key. After your key is created, it will be saved in `/etc/httpd/conf/httpsd.key`, your secure web server's `SSLCertificateKeyFile`.

SSLCACertificateFile The `SSLCACertificateFile` names a file where your secure web server keeps all of the certificates (public keys) for all of the CAs with which it interacts. This file is used for client authentication. Your secure web server's `SSLCACertificateFile` is `/etc/ssl/ca-cert-bundle.pem`.

SSLVerifyClient `SSLVerifyClient` sets a level of required verification for client authentication. By default, your secure web server's `SSLVerifyClient` is set to `none`, meaning that it doesn't require client authentication.

SSLLogFile The `SSLLogFile` provides a place for any error messages generated by SSL. Most of these error messages will be duplicated in your secure web server's "real" error log. By default, your secure web server's SSL-specific error log is `/var/log/httpd/sslstat_log`.

2.3 srm.conf

The srm.conf file defines the server's name space, how requests are serviced and how request results are formatted.

2.3.1 Important Directives in srm.conf

DocumentRoot The DocumentRoot is the directory which contains most of the HTML files which will be served in response to requests. The default DocumentRoot for both the non-secure web server and secure web server is /home/httpd/html. For example, the server might receive a request for the following document:

```
http://www.yourserver.com/foo.html
```

The server will look for the following request in the default directory:

```
/home/httpd/html/foo.html
```

If you want to change the DocumentRoot so that it is no longer shared by the secure web server and the non-secure web server, see section 2.6 on page 36 for instructions.

UserDir UserDir is the name of the sub-directory within each user's home directory where they should place personal HTML files which are to be served by the web server. By default, the subdirectory is public_html. For example, the server might receive the following request:

```
http://www.yourserver.com/~username/foo.html
```

The server would look for the file:

```
/home/username/public_html/foo.html
```

In the above example, /home/username is the user's home directory.

DirectoryIndex The `DirectoryIndex` is the default page served by the server when a user requests an index of a directory by specifying a forward slash (/) at the end of the directory name.

For example, when a user requests the page `http://www.yourserver.com/thisdirectory/`, they are going to get either the `DirectoryIndex` page if it exists, or a server-generated directory listing. The default for `DirectoryIndex` is `index.html index.shtml index.cgi`. The server will try to find any one of these three files, and will return the first one it finds. If it doesn't find any of these files, the server will generate and return an HTML listing of the subdirectories and files in the directory.

Alias The `Alias` setting allows directories to be outside the `DocumentRoot` directory and yet still accessible to the web server. Any URL ending in the alias will automatically resolve to the alias' path. By default, one alias is already set up. An `icons` directory can be accessed by the URL `http://www.yourserver.com/icons/`. The `icons` directory, an alias, is actually `/home/httpd/icons/`, not `/home/httpd/html/icons/`.

ScriptAlias The `ScriptAlias` setting defines where CGI scripts (or other types of scripts) can be found. Generally, you don't want to leave CGI scripts within the `DocumentRoot`. If CGI scripts are in `DocumentRoot`, they are visible as text documents. Even if you don't care if people can see (and then use) your CGI scripts, revealing how they work creates opportunities for unscrupulous people to exploit any security holes in the script, and may create a security risk for your server. By default, the cgi-bin directory is a `ScriptAlias` of `/cgi-bin/`, and is actually located in `/home/httpd/cgi-bin/`.

2.3.2 Other Directives in srm.conf

FancyIndexing `FancyIndexing` affects the appearance of server generated directory listings, by adding icons and descriptions, etc. If `FancyIndexing` is turned on, clicking on the column headers will sort the order of the display by that header. Another click on the same header will switch from ascending to descending order and back. `FancyIndexing` is set to `on` by default.

AddIconByEncoding This directive names icons which will be displayed by files with mime-encoding, in server generated directory listings. For example, by default, your web server shows the `compressed.gif` icon next to mime-encoded `x-compress` and `x-gzip` files in server generated directory listings.

AddIconByType This directive names icons which will be displayed next to files with mime-types in server generated directory listings. For example, your server is set to show the icon `text.gif` next to files with a mime-type of "text," in server generated directory listings.

AddIcon `AddIcon` tells the server which icon to show in server generated directory listings for certain file types or for files with certain extensions. For example, your web server is set to show the icon `binary.gif` for files with `.bin` or `.exe` extensions.

DefaultIcon `DefaultIcon` names the icon to show in server generated directory listings for files which have no other icon specified. `unknown.gif` is the `DefaultIcon` for those files by default.

AddDescription You can use `AddDescription` to show text that you specify for certain files, in server generated directory listings. You can name specific files, wildcard expressions or file extensions to specify the files which this directive should apply to. For example, you could use the following line:

```
AddDescription "A file that ends in .nih" .nih
```

In server generated directory listings, all files with extensions of `.nih` would have the description `A file that ends in .nih` after the filename. Note that you'll also need `FancyIndexing` turned on.

ReadmeName `ReadmeName` names the file which (if it exists) will be appended to the end of server generated directory listings. The web server will first try to include the file as an HTML document and then try to include it as plain text. By default, `ReadmeName` is set to `README`.

HeaderName `HeaderName` names the file which (if it exists) will be prepended to the start of server generated directory listings. Like `ReadmeName`, the server will try to include it as an HTML document if possible, or plain text if not.

IndexIgnore `IndexIgnore` lists file extensions, partial filenames, wildcard expressions or full filenames. The web server will not include any files which match any of those parameters in server generated directory listings.

AccessFileName `AccessFileName` names the file which the server should use for access control information in each directory. By default, your web server is set to use `.htaccess`, if it exists, for access control information in each directory.

TypesConfig `TypesConfig` names the file which sets the default list of mime type mappings (filename extensions to content types). The default `TypesConfig` file is `/etc/mime.types`. Instead of editing `/etc/mime.types`, the recommended way of adding mime type mappings is to use the `AddType` directive.

DefaultType `DefaultType` sets a default content type for the web server to use for documents whose mime types can't be determined. Your web server defaults to assume a plain text content type for any file with an indeterminate content type.

AddEncoding `AddEncoding` names filename extensions which should specify a particular encoding type. `AddEncoding` can also be used to instruct some browsers (not all) to uncompress certain files as they are downloaded.

AddLanguage `AddLanguage` associates filename extensions with specific content languages. This directive is mostly useful for content negotiation, when the server returns one of several documents based on the client's language preference as set in their browser.

LanguagePriority `LanguagePriority` allows you to set precedence for different languages in which to serve files, which will be in effect if the client expressed no preference.

Redirect `Redirect` lets you specify the new URL for documents which have moved from one URL to another. When requests come in for such a document, the new URL will be returned to the client brower, which will attempt to retrieve it automatically.

AddType Use the `AddType` directive to define mime type and file extension pairs. For example, if you are using `mod_php`, your web server is using the `AddType` directive to make your web server recognize files with PHP extensions (`.php3 .phps .phtml`) as PHP mime types.

AddHandler `AddHandler` maps file extensions to specific handlers. For example, the `cgi-script` handler can be used matched with the extension `.cgi` to automatically treat a file ending with `.cgi` as a CGI script. Your web server uses `AddHandler` to process server-parsed HTML and imagemap files.

Action `Action` allows you to specify a mime type and CGI script pair, so that whenever a file of that media type is requested, a particular CGI script will be executed.

MetaDir `MetaDir` specifies the name of a directory where your web server should look for files containing meta information (extra HTTP headers) to include when serving documents.

MetaSuffix `MetaSuffix` specifies the filename suffix for the file that contains meta information (extra HTTP headers), which should be located in the `MetaDir` directory.

ErrorDocument By default, in the event of a problem or error, your web server outputs a simple (and sometimes cryptic) error message back to the client. Instead of using the default, you can use `ErrorDocument` to configure your web server so that it outputs a customized message or redirects to a local or external URL. `ErrorDocument` simply associates a HTTP response code with a message or a URL which will be sent back to the client.

MimeMagicFile `MimeMagicFile` examines a few bytes of a file's contents, in order to determine its mime type. If you want to enable the `MimeMagicFile` directive, you'll also need to load in the `mime_magic_module`. See section 2.5 on page 34 for instructions on how to load in a module to your web server.

BrowserMatch The `BrowserMatch` directive allows your server to define environment variables and/or take appropriate actions based on the `User-Agent` HTTP header field, which identifies the client's browser. By default, your web server uses `BrowserMatch` to deny connections to specific browsers with known problems and also to disable keepalives and HTTP header flushes for browsers that are known to have problems with those actions.

2.4 access.conf

The `access.conf` file contains information about the types of services your server allows, and in what circumstances. You won't need to change any of the default settings in `access.conf` to get your secure web server started and you probably won't need to alter any of its directives for most configurations.

2.4.1 Directives in access.conf

Directory `<Directory /path/to/directory>` and `</Directory>` tags are used to enclose a group of configuration directives that are meant to apply only to that directory and all of its subdirectories. Any directive which is applicable to a directory may be used within `<Directory>` tags.

By default, your `access.conf` file is set up to apply very restrictive parameters for your root directory, using the `Options` and `AllowOverride` directives. Under this configuration, any directory on your system which needs more options has to be explicitly given those options.

Two directories are defined to have less rigid parameters: `/home/httpd/html` and `/home/httpd/cgi-bin`.

Options The `Options` directive controls which server features are available in a particular directory. For example, under the restrictive parameters specified for the root directory in `access.conf`, `Options` is set to `None`. No features are enabled.

By default, in your `/home/httpd/html` directory, `Options` is set to include `Indexes`, `FollowSymLinks` and `Includes`. `Indexes` permits the server to generate a directory listing for a directory if no `DirectoryIndex` (i.e., `index.html`) is specified. `FollowSymLinks` allows the server to follow symbolic links in that directory. `Includes` means that server-side includes are permitted.

Your /home/httpd/cgi-bin directory has Options ExecCGI set, meaning that execution of CGI scripts is permitted within that directory.

AllowOverride The AllowOverride directive sets whether or not any Options can be overridden by the declarations in a .htaccess file. By default, all of the directories set up in your access.conf are set up to allow no .htaccess overrides.

order The order directive simply controls the order in which allow and deny directives are evaluated. Your server is configured to evaluate the allow directives before the deny directives for your /home/httpd/html directory.

allow allow specifies which requester can access a given directory. The requester can be all, a domain name, an IP address, a partial IP address, a network/netmask pair, etc. Your /home/httpd/html directory is configured to allow requests from all (i.e., anyone).

Location <Location> and </Location> tags allow you to specify access control based on the URL. For example, by default your web server allows a <Location> directive which redirects any requests which end in /cgi-bin/phf* to a logging cgi run by the Apache group.

Another <Location> option is commented out in your access.conf file. If you want to allow people connecting from your domain to see server status reports, you should uncomment the following lines:

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

You also need to replace .yourdomain.com with your second level domain name.

2.5 Adding Modules to Your Server

Since Apache 1.3 supports Dynamic Shared Objects (DSOs), you can easily load Apache modules or compile in your own modules to your secure web server. DSO support means that modules may now be loaded at runtime. Since the modules are only loaded as necessary, they won't use any memory unless they're loaded and less memory will be needed overall. After installation of your server, check `http://www.yourdomain.com/manual/mod/` for documentation on Apache modules in HTML format.

For your secure web server to use a dynamically shared module, that module must have a `LoadModule` line and an `AddModule` line in `httpd.conf`. By default, many modules have these two lines already included in `httpd.conf`, but a few of the less commonly used modules are commented out. The commented out modules were compiled in during compilation, but they are not loaded by default.

If you need to use one of those non-loaded modules, look in the `httpd.conf` file to see all the available modules. Each of the available modules has a corresponding `LoadModule` line. To show you an example, the `LoadModule` section begins with these six lines:

```
#LoadModule mmap_static_module lib/apache/mod_mmap_static.so
LoadModule env_module lib/apache/mod_env.so
LoadModule config_log_module lib/apache/mod_log_config.so
LoadModule agent_log_module lib/apache/mod_log_agent.so
LoadModule referer_log_module lib/apache/mod_log_referer.so
#LoadModule mime_magic_module lib/apache/mod_mime_magic.so
```

Most of the lines are not commented out, indicating that each associated module was compiled in and is loaded in by default. The first line is commented out, which means that the corresponding module (`mmap_static_module`) was compiled in but not loaded.

To make your secure web server load in an unloaded module, first uncomment the corresponding `LoadModule` line. For example, if you wanted to make your secure web server load in the `mime_magic_module`, change that `LoadModule` line from the original:

```
#LoadModule mime_magic_module lib/apache/mod_mime_magic.so
```

Uncomment the previous line so that it reads:

```
LoadModule mime_magic_module lib/apache/mod_mime_magic.so
```

Next, you need to uncomment the corresponding line from the `AddModule` section in `httpd.conf`. To continue with our previous example, uncomment the `mod_mime_magic` line. The original (default) line looks like the following:

```
#AddModule mod_mime_magic.c
```

The uncommented line should read:

```
AddModule mod_mime_magic.c
```

Once you've uncommented the `LoadModule` and `AddModule` lines for the module that you want to load in, stop and start your web server, as covered in section 2.7 on page 40. After starting, the module should be loaded in to your secure web server.

If you have your own module, you can add it to the `httpd.conf` file so that it is compiled in and loaded as a DSO. If you want to do this, you need to have installed the `devel` package during the installation process, as covered in section 1.4 on page 9. You need the `devel` package because it installs the include files, the header files and the APache eXtenSion (APXS) support tool. APXS uses the include files and the header files to compile your module so that it will work with Apache.

If you've written your own module or are borrowing someone else's, you should be able to use APXS to compile your home-made module sources outside the Apache source tree, without needing to tweak any compiler and/or linker flags. If you need more information on APXS, please see the Apache documentation at `http://www.apache.org/docs/dso.html`.

Once you've compiled your module using APXS, put your module into `/usr/lib/apache`. Then your module needs both a `LoadModule` line and an `AddModule` line in the `httpd.conf` file, just as described previously for Apache's own modules.

After the `LoadModule` section, at the end of the `Additional modules:` section in `httpd.conf` (after the mod_php and mod_perl modules), add a line for the shared object file for your module like the following:

```
LoadModule foo_module  lib/apache/mod_foo.so
```

Note that you'll need to change the name of the module and the name of your shared object file as appropriate.

At the end of the `Additional Modules:` section after the `AddModule` section in `httpd.conf` (after the lines for the mod_php and mod_perl modules), add a line for the source code file for your module like the following:

```
AddModule mod_foo.c
```

Note that you'll need to change the name of the source code file as appropriate.

Once you've completed the previous steps, stop and start your web server as outlined in section 2.7 on page 40. If you've done everything correctly, and your module is correctly coded, your web server should find your module and load it in as it starts.

2.6 Using Virtual Hosts

You can use Apache's virtual hosts capability to run different servers for different IP addresses, different host names or different ports on the same machine. If you're interested in using virtual hosts for different IP addresses or different host names on your machine, more information is provided in the Apache documentation on your machine or on the web at `http://www.apache.org/docs/vhosts/index.html`.

Virtual hosts are configured within the `httpd.conf` file, as described in section 2.2 on page 17. Please review the pertinent points in that section before you start to change the virtual hosts configuration on your machine.

The default configuration of your secure web server uses virtual hosts to run two web servers: a secure web server and a non-secure web server. Both servers use the same IP address and host name, but they listen on different ports. This configuration enables you to serve both secure and non-secure documents in the most efficient manner possible. As you may know, secure HTTP transmissions take more time than non-secure, because a lot more information is being passed back and forth during the secure transaction. So using a secure web server for non-secure web traffic is not a good idea.

As installed, your secure web server is configured to operate in both secure and non-secure modes. The non-secure web server is configured as the "non-virtual" host in the `httpd.conf` file. In other words, the non-secure web server's configuration options are outside of the virtual host tags in `httpd.conf`. If you want to change something about your non-secure web server, you'll need to change the configuration directives in `httpd.conf` outside of the virtual host tags. Your non-secure web server, by default, listens on port 80, the default port for non-secure web communications.

Your secure web server is configured as a virtual host in `httpd.conf`. The configuration options for your secure web server are contained within the virtual host tags in the `httpd.conf` file. If you need to change something about the configuration of your secure web server, you'll need to change the configuration directives inside the virtual host tags in the `httpd.conf` file. Your secure web server, by default, listens on port 443, the default port for secure web communications.

By default, the secure and the non-secure web server share the same `DocumentRoot`, a configuration directive specified in `httpd.conf`. In other words, the secure and the non-secure web server look in the same place for the HTML files that they are going to provide in response to requests. By default, the `DocumentRoot` is set to `/home/httpd/html`.

To change the `DocumentRoot` so that it is no longer shared by both the secure server and the non-secure server, change one of the `DocumentRoot` directives in either `httpd.conf` or in `srm.conf`. The `DocumentRoot` in `srm.conf` defines the `DocumentRoot` for your non-secure web server. The `DocumentRoot` in `httpd.conf` (note that it is located within the virtual host tags that define your secure web server) is for your secure web server.

Obviously, if you change the `DocumentRoot`, make sure that the new one is accessible to the `User`.

If for some reason you want to disable the non-secure web server on your machine, you can. In `httpd.conf`, change the line which reads:

```
Port 80
```

so that it reads:

```
Port 443
```

Then comment out the `Listen 80` line, so that instead of:

```
Listen 80
```

it reads:

```
#Listen 80
```

After these two steps, your secure web server will be accepting connections on port 443, the default port for secure web communications. However, your server will not accept connections on port 80, the default port for non-secure communications, so the non-secure web server will be effectively disabled.

Most people will probably use their secure web server as it is configured. Therefore, they'll be using the built-in virtual hosts capability, but they won't have to do any manipulation of the virtual hosts directives in `httpd.conf`. However, if you would like to use the virtual hosts capability for some other reason, you can.

To create a new virtual host, you'll need to alter the virtual host lines in `httpd.conf`. Find the lines which read as follows:

```
#<VirtualHost host.some_domain.com>
#ServerAdmin webmaster@host.some_domain.com
#DocumentRoot /www/doc/host.some_domain.com
#ServerName my.domain.com
#ErrorLog logs/host.some_domain.com-error_log
#TransferLog logs/host.some_domain.com-access_log
#</VirtualHost>
```

Uncomment all of the lines (remove the # from the beginning of each line). Then add the correct information for your machine and/or your virtual host to each line.

For example, in the first line, `<VirtualHost host.some_domain.com>`, change the server's name to whatever your server's name is. You'll need to use the same name for both your virtual server and your "real" server, unless you have another *valid* DNS name to use for the virtual host. In other words, don't just make something up. Ask your system administrator if you don't know how to get a valid domain name.

Many other configuration directives can be placed between the virtual host tags, depending upon why you're setting up a virtual host.

If you set up a virtual host and want it to listen on a non-default port (80 is the default port for non-secure web communications; 443 is the default port for secure web communications), you'll need to set up a virtual host for that port and add a `Listen` line to `httpd.conf` corresponding to that port.

To have a virtual host work specifically for that port, add the port number to the first line of the virtual host configuration. The first line should look something like the following:

```
<VirtualHost www.yourserver.com:12331>
```

This line would create a virtual host that listens on port 12331. Substitute the port number you want to use for `12331` in the previous example.

Underneath the `Listen` lines in `httpd.conf`, add a line like the following:

```
Listen 12331
```

This line instructs your web server to listen on port 12331.

You must restart your server to start your virtual host. However, if you have not yet created a digital certificate as discussed in Chapter 3 on page 43, your secure web server will not work.

Much more information about creating and configuring virtual hosts is provided on the web at
`http://www.apache.org/docs/vhosts/index.html`.

2.7 Starting and Stopping Your Server

During the installation process, a server startup file named `httpsd` was installed into `/etc/rc.d/init.d`. To manually stop and start the server, run this `httpsd` command with either `stop` or `start` as an argument.

For example, to stop your server, type the command:

```
/etc/rc.d/init.d/httpsd stop
```

To start your server, type the command:

```
/etc/rc.d/init.d/httpsd start
```

You will be prompted to fill in your password. After you type it in, your server will start.

You may also use the command `restart`, which is a short way of stopping and then starting your server. `restart` does explicitly stop and then start your server, so you will be prompted for your password. `restart` looks like the following:

```
/etc/rc.d/init.d/httpsd restart
```

If you just finished editing something in your configuration files (`httpd.conf`, `srm.conf` or `access.conf`), you don't need to explicitly stop and start your server. Instead you may use the `reload` command. The benefit of using `reload` is that you will not need to type in your password. Your password will remain cached across reloads, but it will not be cached betweens stops and starts. `reload` should look like the following:

```
/etc/rc.d/init.d/httpsd reload
```

After a reload, your server will start without prompting you for your password.

If you want your server to start automatically when your machine boots, you need to create symlinks from the `rc3.d` and `rc0.d` directories to the `httpsd` command. Use the following commands create the symlinks:

```
cd /etc/rc.d/rc3.d
ln -s ../init.d/httpsd S99httpsd
```

The previous command tells your system to start the `httpsd` process when it boots.

```
cd /etc/rc.d/rc0.d
ln -s ../init.d/httpsd K99httpsd
```

The previous command tells your system to stop the `httpsd` process when the system is shut down.

If you are running the X Window System, you can use `tksysv` to create these symlinks.

2.8 Accessing Your Server

The standard port for secure web communications is port 443. The standard port for non-secure web communications is port 80. After you've configured your machine according to the guidelines in this chapter, it will be set up to listen on one or both of the two standard ports. Therefore, you won't need to specify the port number in a URL.

However, if you've configured your server to listen on a non-standard port (i.e., anything besides 80 or 443), you'll need to specify the port number in every URL which is intended to connect to the server on the non-standard port.

For example, you may have configured your server so that you have a virtual host running non-secured on port 12331. Any URLs intended to connect to that virtual host must specify the port number in the URL. The following URL example will attempt to connect to a non-secure web server listening on port 12331:

```
http://www.yourserver.com:12331
```

You should also remember that URLs which are intended to connect to your secure web server should begin with the `https:` protocol designator instead of the more common `http:` protocol designator. `https:` is the protocol designator for secure HTTP protocol communications.

Some of the example URLs used in this manual may need to be changed, depending upon whether you're accessing your secure web server or your non-secure web server. Please view all URLs in this manual as general examples, not as explicit instructions that will work under all circumstances.

3

Securing Your Server

Since you purchased this product, you are probably interested in conducting electronic commerce using your web site. To make your customers feel safe doing business with you over the web, your web server needs to be secure. A secure server instructs any browser accessing your site to encrypt all the data sent back and forth. Encrypted data can't be intercepted and read by a third party. Your customers will feel more comfortable when making purchases from your site if they know that their transactions are secure.

Secure servers aren't only for electronic commerce. A secure server may also be used to transmit sensitive data such as sales figures to sales people on the road or to business partners over the Internet.

This chapter will show you how to create a key and a certificate request. Then we'll go through the process of getting a certificate signed by one of two different Certificate Authorities (CAs): VeriSign (`http://www.verisign.com`) and Thawte (`http://www.thawte.com`). Once you have a signed certificate, you'll learn how to install it on your secure web server.

When you use a signed certificate, you guarantee the identity of the organization running the server. For example, if the certificate says the website is Red Hat Software's, and the user trusts the CA, then there is no reason to doubt that any files or programs downloaded from that site really are from Red Hat Software, Inc.

If you upgraded from the original Red Hat Secure Web Server (version 1.0), your old key (`httpsd.key`) and certificate (`httpsd.crt`) will be located in `/etc/httpsd/conf`. Move these two files to `/etc/httpd/conf`. Then start your secure web server as described in section 2.7 on page 40. You should not need to get a new certificate.

Please Note: VeriSign is a very widely used CA. If you already have a VeriSign certificate for another purpose or that you used for a different web server software, you may have been considering using your existing VeriSign certificate with your new secure web server. However, you are not allowed to, because VeriSign issues certificates for one particular server software and IP address/domain name combination.

If you change either of those parameters (for example, if you previously used another secure web server product and now you want to use the Red Hat Secure Web Server), the VeriSign certificate you obtained to use with the previous configuration will not work with the new configuration. You will have to obtain a new VeriSign certificate.

3.1 How Server Security Works

Internet security depends upon the successful interaction and outcome of three factors: authentication, integrity and privacy. Authentication is when you know with certainty the identity of the person or organization with whom you are communicating. Integrity is when the data you send is the data that is received (i.e., no tampering with the data occurred on its way to its destination). Privacy is when no third party can intercept and understand the private communication.

Your Red Hat Linux Secure Web Server provides for these factors by incorporating SSLeay (an implementation of SSL, the Secure Sockets Layer protocol) into the Apache web server and by the use of CA-approved digital certificates. SSL handles the encrypted communications (integrity and privacy) and the mutual authentication between browsers and your secure web server. The CA-approved digital certificate provides authentication (the CA puts its reputation behind a certification of your organization's identity).

Encryption depends upon the use of keys (think of them as secret encoder/decoder rings in data format). In conventional or symmetric cryptography, both ends of the transaction have the same key, which they use to decode each other's transmissions. In public or asymmetric cryptography, two keys co-exist: a public key and a private key. A person or an organization keeps their private key a secret, and publishes their public key. Data encoded with the public key can only be decoded with the private key; data encoded with the private key can only be decoded with the public key.

You'll use public cryptography to create a public and private key pair. Then you'll create a certificate request, which contains your public key. You send your certificate request, proof of your company's identity and payment to a CA. The CA verifies the certificate request and your identity, and then sends back a certificate for your secure web server.

Your secure web server needs a certificate signed by a CA, so that people who visit your website know you are who you claim to be. When a web browser connects to a secure website, the browser requests that the server provide some evidence of its identity (or more importantly, the identity of the organization behind the website). The certificate is signed by a CA, which basically means that you can be assured of its validity. Before signing the certificate, the CA verified that the organization (in this case, you) requesting the certificate was actually who they claimed to be.

Most web browsers that support SSL have a list of CAs whose certificates they will automatically accept. If a browser encounters a certificate whose authorizing CA is not in the list, the browser will ask the user to choose whether to accept or decline the connection.

The process of getting a certificate is relatively easy and this chapter will cover how to get one in detail. A quick overview is as follows: you create an encryption key and then a certificate request based on that key. The certificate request contains information about your server and the company hosting it. You send this certificate request, along with documents proving your identity, to a CA. Once the CA is satisfied that you are indeed who you claim to be, they will send you a digital certificate. You then install this certificate on your web server, and begin handling secure transactions.

3.2 Deciding on a Certificate Authority

We can't tell you which certificate authority to choose. Your decision may be based on your past experiences, or on the experiences of your friends or colleagues, or purely on monetary factors. We will guide you through the process of getting a digital certificate from both VeriSign and Thawte.

At the time this document was written, VeriSign charged $349 for the first server digital certificate and $249 for each additional server. Thawte charged $125 for each server. Each of these amounts will get you a certificate good for a limited amount of time, normally a year. You'll need to check with the CA when you purchase the certificate to find out exactly when your certificate will need to be renewed.

Please Note: Through a special arrangement with Thawte, purchasers of the Red Hat Secure Web Server boxed set can obtain signed certificates for only $100 per server. Instructions for obtaining the discounted price can be found in section 3.7.2 on page 61.

3.3 Proving Your Organization's Identity to a CA

When you request a certificate from a CA, you'll need to somehow prove that your organization has the right to conduct business using your organization's name. CAs are very specific about what types of proof they will accept.

In some cases, copies of the documents described will need to be mailed or faxed to the CA, and your certificate will not be issued until the documents have been received and verified by the CA.

3.3.1 Proving Your Organization's Identity to VeriSign

The easiest way to prove to VeriSign that your organization has the right to do business is to provide your Dun & Bradstreet (D-U-N-S) number. If you don't have a D-U-N-S number, you can request one from the Dun & Bradstreet website at `http://www.dnb.com/aboutdb/dunsform.htm`.

If you don't know whether you have a D-U-N-S number, you can find out if you have one from VeriSign at
`https://digitalid.verisign.com/dnb_query.htm`.

If you don't want to use a D-U-N-S number and the certificate is for a business, you can also provide a copy of any one of the following documents to VeriSign instead:

1. Articles of Incorporation
2. Business License
3. Fictitious Business License
4. Official Filing with the Security & Exchange Commission

If you don't want to use a D-U-N-S number or you don't have a D-U-N-S number and the certificate is for a government organization or agency, you need to supply VeriSign with a copy of the specific page of the Act or Charter that established your organization within the government.

If you don't want to use a D-U-N-S number or you don't have a D-U-N-S number and the certificate is for a university, you can provide VeriSign with a copy of your university's Articles of Incorporation.

If your organization doesn't fit into any of these categories, or if you need more information on what your organization can provide VeriSign to prove your right to conduct business under your own name, see VeriSign's website at `http://www.verisign.com`. You should check VeriSign's website in any case, just to make sure that these requirements haven't changed since this document was produced.

3.3.2 Proving Your Organization's Identity to Thawte

Thawte requires some form of all three of the following to prove your organization's identity:

1. Proof of the Right to a Name
2. Proof of the Right to a Domain
3. Letter of Authorization for Certificate

Proof of the right to a name consists of some type of documentation that proves you are a recognized business or non-profit known by a particular name. The type of documentation you'll need to provide depends upon what kind of organization your company is (for example, a corporation, a government department, a university, etc.).

If your organization is a company, corporation, partnership or proprietorship, you'll need to provide Thawte with either a copy of your company registration documents or a copy of your articles of incorporation/partnership declaration.

If your organization is a DBA (Doing Business As), you'll need to provide Thawte with a copy of your DBA registration papers for local levies and taxes.

If your organization is a government department, you'll need to provide Thawte with an original signed letter from the head of your department, providing contact information for him or her and for his or her immediate superiors.

If your organization is an NGO (Non-Government Organization), you'll need to provide Thawte with an original signed letter from the Chief Executive, Chairman or Managing Director of your organization on letterhead.

If your organization is a university or university department, you'll need to provide Thawte with an original signed letter on letterhead from the Dean or head of the department, including contact information for him or her.

If your organization is some kind of special interest group or just doesn't fit into any of the above categories, you'll need to check Thawte's website at http://www.thawte.com for more information or contact Thawte at server-certs@thawte.com to ask them what documents you'll need to provide.

Proof of the right to a domain consists of information proving that you registered for your domain name. You won't need to provide this proof if your domain name ends in .com, .edu, .org, .se, or .za.

Proof can be a copy of the domain registration application made to InterNIC, the organization that handles all domain registrations within the US, or a whois query output for your domain name. Thawte will want to know which authority you registered with (usually it will in InterNIC) and the names and contact information for the person or persons who were named in the domain name registration application as administrative contacts and/or technical contacts.

The **letter of authorization** is a signed letter from a high-ranking member of your organization, signifying that the organization's decision makers approve of the use of a Thawte digital certificate to authenticate the company during secure online transactions. This letter is supposed to prevent non-authorized people from creating an authenticated electronic presence without first going through the proper channels.

Please visit Thawte's web site to make sure these requirements haven't changed. This list is meant as a general overview and is not guaranteed to be complete.

See Thawte's website at http://www.thawte.com for more guidance on what kind of documentation you'll need to provide.

Once you've gathered the materials you'll need to submit, section 3.7 on page 59 provides the complete steps for requesting a certificate from these certificate authorities. Before you request a certificate, however, you'll need to generate a key and a certificate request.

3.4 Creating Your Key and Certificate Request

You'll use the SSLeay program, which was installed with your secure server, to generate an encryption key and a certificate request. You need the key to create the certificate request. You need the certificate request in order to apply for a certificate from a CA. Finally, you need the certificate to run a secure web server.

3.4.1 Generating a Key

First, you'll use SSLeay and the system file `/dev/urandom` to generate a random key. `cd` to the `/etc/httpd/conf` directory. Type in the following command, which will generate a 1024 bit key encrypted with the triple-DES cipher:

```
make genkey
```

If, for some reason you do not have `make` installed on your system, you may use the following less user-friendly command *instead* of `make genkey`:

```
/usr/sbin/ssleay genrsa -des3 -rand /dev/urandom 1024 > httpsd.key
```

Your system will display a message similar to the following:

```
1049776 semi-random bytes loaded
Generating RSA private key, 1024 bit long modulus
.........................................+++++
.............+++++
e is 65537 (0x10001)
Enter PEM pass phrase:
```

You now need to type in a password. For best security, your password should be at least eight characters, should include numbers or punctuation, and should not be a word in a dictionary. Also, remember that your password is case sensitive. You will need to remember and enter this password every time you start your secure web server, so don't forget it.

You will be asked to re-type the password, just to make sure that you didn't make any typos entering it. Once you've typed it in correctly, a file called `httpsd.key`, containing your key, will be created.

Please Note: If you don't want to have to type in a password every time you start your web server, you will need to use the long command as shown above without the `-des3` option:

```
/usr/sbin/ssleay genrsa -rand /dev/urandom 1024 > httpsd.key
```

If you use the immediately previous command to create your key, you will not need to use a password to start your secure web server. Please realize, however, that disabling the password feature for your secure web server is a security risk. We DO NOT recommend that you disable the password feature for your secure web server.

The `httpsd.key` file should be owned by the root user on your system and should not be accessible to any other user. Make a backup copy of this file and keep the backup copy in a safe, secure place (a good idea would be to copy this file to a floppy and then keep the floppy someplace secure). You need the backup copy because if you ever lose the `httpsd.key` file after using it to create your certificate request, your certificate will no longer work and the CA will not be able to help you. Your only option would be to request a new digital certificate and pay for it all over again.

3.4.2 Generating a Certificate Request

Once you've created a key, the next step is to generate a certificate request which you will need to send to the CA of your choice. Type in the following command:

```
make certreq
```

If you don't have `make` installed on your system, you may use the following longer and less user-friendly command instead of `make certreq`:

```
ssleay req -new -key httpsd.key > httpsd.csr
```

Your system will display the following output and will ask you for your password (if you disabled the password option, it won't ask):

```
Using configuration from /etc/ssl/lib/ssleay.cnf
Enter PEM pass phrase:
```

Type in the password that you chose when you were generating your key. Your system will display some instructions and then ask for a series of inputs from you. Your inputs will be incorporated into the certificate request. The display, with example inputs, will look like this:

```
You are about to be asked to enter information
that will be incorporated into your certificate
request.
What you are about to enter is what is called a
Distinguished Name or a DN.
There are quite a few fields but you can leave
some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name []:North Carolina
Locality (City) Name []:Durham
Company (Organization) Name []:Test Company
Department Name []:Testing
Server Host Name []:test.mydomain.com
Administrators E-mail address []:admin@mydomain.com
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

The default answers appear in brackets `[]` immediately after each request for input. For example, the first information required is the name of the country where the certificate will be used, shown like the following:

```
Country Name (2 letter code) [US]:
```

Since the Red Hat Secure Web Server is restricted for sale to only the US or Canada, your input will be either `US` or `CA`. The default input, in brackets, is `US`. To accept the default input, just press Enter.

You will have to type in the rest of the inputs (`State or Province Name`, `Locality (City) Name`, `Company (Organization) Name`, `Department Name`, `Server Host Name`, and `Administrators E-mail address`). All of these should be self-explanatory. Do not abbreviate the city or state. Write them out (for example, St. Louis should be written out as Saint Louis). For `Server Host Name`, make sure you type in the *real* name of your secure web server (a valid DNS name), and not any aliases which the server may have.

You don't need to use either of the extra attributes (`A challenge password` and `An optional company name`). Just press Enter to accept the blank default for both inputs.

When you've finished inputting your information, a file named `httpsd.csr` will be created. `httpsd.csr` is your certificate request, ready to send to your CA.

Next, you need to test your secure web server to make sure everything is working properly. To do that, you need to either send the certificate request off to a CA in order to get a test certificate, or to create a test certificate yourself.

Note that a *test* certificate lets you test your secure web server to see if it works, but it is not the same thing as a signed certificate. A test certificate can be used for a short period of time and you don't have to pay for it.

However, it is illegal to use a test certificate as if it were a real certificate (i.e., to engage in Internet commerce with it) and it won't work. A test certificate will not be accepted by browsers which would normally accept that CA's certificates without asking the user. Test certificates will only work with browsers which have been equipped with a root certificate provided with the test certificate. You will have to insert the root certificate into a browser or browsers in order to test your secure web server with those particular browsers.

You are only allowed to use the test certificate for testing or demonstration purposes. You're not allowed to use the test certificate for secure commerce over the web.

The next section covers how to get test certificates from both VeriSign and Thawte, as well as how to create a test certificate yourself.

3.5 Getting a Test Certificate

Both VeriSign and Thawte will issue you a temporary certificate which you can use to test your secure web server to make sure it is working. VeriSign's test certificate lasts for two weeks while Thawte's lasts for thirty days.

Another option, which we'll discuss first, is to create your own test certificate using the SSLeay library.

3.5.1 Creating a Test Certificate On Your Own

To create your own test certificate, type in `make testcert` or the following command:

```
ssleay req -new -x509 -key httpsd.key > httpsd.crt
```

Your display will show the following text and wait for your input. Note that this is very similar to the input you typed in when you generated your key. You can press Enter to accept the default input, which is shown within brackets. Again, remember not to abbreviate anything. If you disabled the password option, it will not prompt you for a password.

```
Using configuration from /etc/ssl/lib/ssleay.cnf
Enter PEM pass phrase:
You are about to be asked to enter information
that will be incorporated into your certificate
request.
What you are about to enter is what is called a
Distinguished Name or a DN.
There are quite a few fields but you can leave
some blank
For some fields there will be a default value,
```

```
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name []:North Carolina
Locality (City) Name []:Durham
Company (Organization) Name []:Test Company
Department Name []:Testing
Server Host Name []:test.mydomain.com
Administrators E-mail address []:admin@mydomain.com
```

Once you've typed in all of the required information, SSLeay will generate a test certificate and place it in a file named `httpsd.crt`. You can use `httpsd.crt` to test your secure server.

See section 3.6 on page 58 to install this test certificate and test your secure web server.

3.5.2 Obtaining a Test Certificate from VeriSign

To request a test certificate from VeriSign, point your web browser to `https://digitalid.verisign.com/server/trial/index.html`, an informational page about test certificates. Follow these instructions:

1. Read through the information on the first page, entitled "Test Drive a Secure Server ID." When you are finished, click the **Enroll Now** button in the bottom right corner.

2. Read through the information on the "Before You Start" page. You'll probably want to take a look at the VeriSign Trial Subscriber Agreement. Click on **Continue** when you're finished.

3. The next page, "Generate CSR," shows you instructions on how to generate a CSR. If you followed the instructions in section 3.4.2 on page 51, you already have a CSR in your `httpsd.csr` file. If not, create your key and CSR using those instructions now. Then click on **Continue** in the bottom right corner to continue.

4. The next page, "Submit CSR," is shown in figure 3.1 on the next page. On this page, you'll actually need to do something. Copy the

contents of your `httpsd.csr` file (including the BEGIN CERTIFICATE REQUEST line and the END CERTIFICATE REQUEST line) and paste them into the text box. After doing so, the page should look something like 3.2 on page 64.

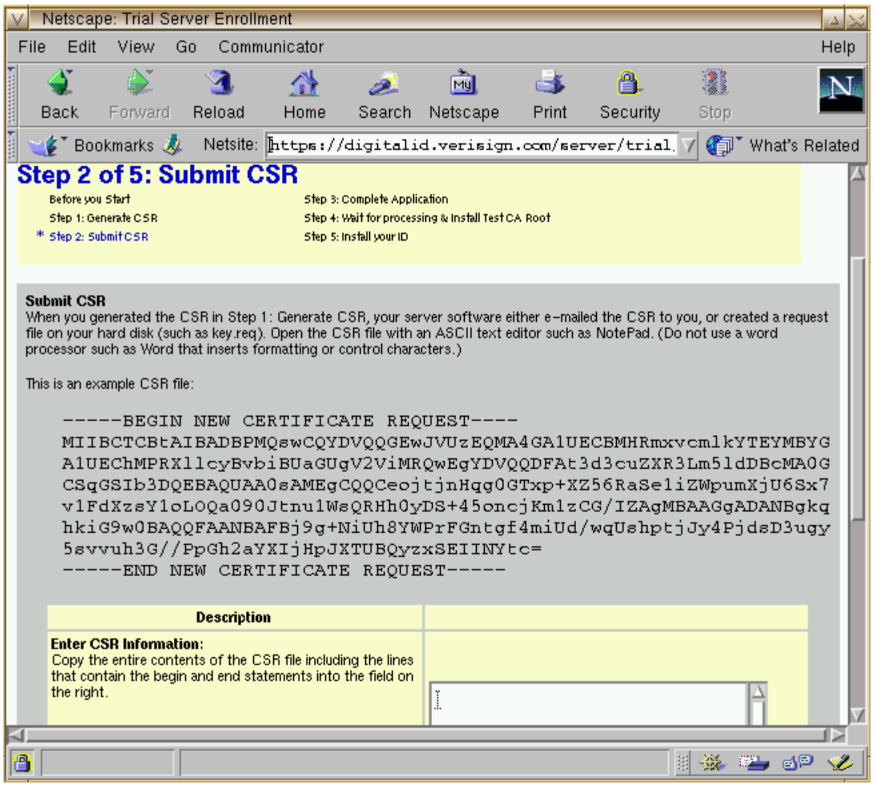

Figure 3.1: Submit CSR

5. Click on **Continue** at the bottom right of the “Submit CSR” page to continue.

6. The next page, “Complete Application,” is shown in figure 3.3 on page 65. The top of this page will be automatically filled in with the information you provided when you created your certificate request.

7. Scroll down to the bottom of the page, which will contain a form for you to fill out. You need to fill in the blanks with contact information for your secure web server's administrator.

8. After you have filled in the correct information for your secure web server's administrator or webmaster, you should read the server agreement provided at the bottom of the page. Then click on **Accept** at the bottom of the page to indicate that you accept the terms of the agreement and that you would like to be issued a test certificate.

   VeriSign should indicate that your application was received and processed and email you a test certificate. If you receive an error message instead, follow whatever instructions it provides, but first check your email. Even if you received an error message indicating that there was some problem with your application and that you'll need to email your application to them, the test certificate may have already been emailed to you.

9. Save the test certificate, including the BEGIN CERTIFICATE line and the END CERTIFICATE line, in a file named `httpsd.crt` in your `/etc/httpd/conf` directory.

See section 3.6 on the following page for instructions on how to install your test certificate.

3.5.3 Obtaining a Test Certificate from Thawte

To request a test certificate from Thawte, point your web browser to `https://www.thawte.com/cgi/server/test.exe` (figure 3.4 on page 66) and follow the instructions provided next.

1. Cut and paste your certificate request (the contents of your `httpsd.csr` file) into the "Certificate Signing Request" text box on Thawte's web page.

2. Select **Use the most basic format** in the "Certificate Format Options" section of the page.

3. Click the **Generate Test Certificate** button at the bottom of the page.

4. The returned web page will include your test certificate (an example is shown as figure 3.5 on page 67). Cut and paste the test certificate into a file named `httpsd.crt` in your `/etc/httpd/conf` directory.

See section 3.6 for instructions on how to install the test certificate and test your secure web server.

3.6 Installing and Testing Your Certificate

At this point, you should have a file named `httpsd.key`, containing your key, and a file named `httpsd.crt`, containing your test certificate. Both of the `httpsd.key` and `httpsd.crt` files should be in your `/etc/httpd/conf/` directory. If they are somewhere else, move them to that directory. If you changed any of the default locations or filenames for the secure web server in your Apache configuration files, you should put these two files in the appropriate directory, based on your modifications.

Now start your server as described in section 2.7 on page 40. If your key file is encrypted, you will be asked for the password. Type in your password. Your server will now be running.

Point your web browser to your server's home page. You should see a dialog box which indicates that your browser needs to be configured to accept the test certificate.

Follow the instructions provided by your browser to accept the test certificate. You can just accept the defaults by clicking **Next** until the dialogs are finished.

Once you've configured your browser to accept the text certificate, your secure web server will show you a default home page as shown in figure 3.6 on page 68.

Your web server is running. The next step is to buy and install a real certificate.

3.7 Buying a Certificate

Now you're ready to purchase a certificate. Once you've received the certificate, simply follow the steps in section 3.6 on the facing page to install it.

If you purchase your certificate from a well-known CA, you probably won't need to modify your browser to accept the certificate, since many browsers are configured to accept well-known CA's signed certificates. However, this depends upon each browser's configuration.

3.7.1 Purchasing a Certificate From VeriSign

Purchasing a Certificate from VeriSign is similar to requesting the free test certificate. Point your browser to `http://digitalid.verisign.com/server/enrollIntro.htm`, which is an informational page about purchasing a certificate.

1. Read the information provided. You'll probably want to print out a copy of their Secure Server Enrollment Guide as suggested. Click on the **Continue** button at the bottom of the page.
2. The next page is entitled "Confirm Domain Name" and is shown as figure 3.7 on page 69. You need a valid, registered domain name in order to obtain a certificate. This page instructs you to obtain a `whois` query result for your domain name. The `whois` result will be used by VeriSign to confirm that your organization actually controls the domain for which you'll be using their certificate.

   Click on whichever link on the right is applicable to you, and use their service to look up a `whois` for your domain name. Note that you'll need to look up the second level domain name (i.e., just `whatever.com` instead of `www.whatever.com`). Print out or save the `whois` query result and use your back button to return to the VeriSign page. Click **Continue**.
3. The next step is to "Obtain Proof of Right." This means that you need to prove to VeriSign that your organization is legitimate. The easiest way to do this is to provide them with your D-U-N-S number,

but there are other ways if you don't have a D-U-N-S number or you don't want to use one. Refer to 3.3.1 on page 47 or to the instructions provided by VeriSign. You'll need that proof, ready for submission to VeriSign, before you can apply for a certificate. Once you have the required documents, press **Continue** at the bottom of the page.

4. The next page is entitled "Generate CSR." If you followed the instructions provided in sections 3.4.1 on page 50 and 3.4.2 on page 51, you have already generated a CSR. Your CSR should be saved as `httpsd.csr` in `/etc/httpd/conf`. If you have not generated a key and a certificate request, follow the instructions in those two sections. Then click on the **Continue** button at the bottom of this page.

5. On the "Submit CSR" page, shown as 3.8 on page 70, you'll see an example of a certificate request.

   Paste the contents of your `httpsd.csr` file into the "Enter CSR Information:" text box. Then press the **Continue** button at the bottom of the page.

6. The next page, "Complete Application," contains an application for you to fill out. The top section ("Verify Distinguished Name") will already be completed from the contents of your certificate request. Scroll down to the next section, "Enter Server ID Information" as shown in figure 3.9 on page 71.

7. Select **Apache Freeware with SSLeay** from the "Select Server Type" pull-down selection box.

8. Type a "challenge phrase" into the area provided. You may be asked for your challenge phrase if you ever need support from VeriSign, so be sure to record it and keep it someplace safe.

9. Fill in the "Enter Technical Contact Information" section with information about your secure web server's administrator or webmaster.

10. Fill in the "Enter Organizational Contact Information" section with information about your secure web server's administrator or webmaster, or whoever you feel is appropriate according to the instructions provided by VeriSign.

11. Fill in the “Enter Billing Contact Information” with information for the person who will be contacted for billing purposes.

12. Select whether this is your first VeriSign certificate or not. You get a discounted price if you already have at least one.

13. Indicate how you are going to pay for your certificate.

14. Fill in your D-U-N-S number. If you don’t have one, you can apply for one at this point. If you are going to use other documents to prove your organization’s legitimacy, you’ll need to fax or mail them to VeriSign, using the instructions provided in their “Secure Server Enrollment Guide,” after you submit your enrollment form.

15. After you’ve read their server agreement, click on the **Continue** button at the bottom of the page. Your application will be submitted.

After you’ve successfully completed your enrollment form and your information and payment has been provided to VeriSign, they will need three to five working days to authenticate your organization’s identity and issue your certificate. When your application has been approved, they will send your certificate by email to the technical and organizational contacts you provided.

Save the certificate VeriSign emails to you in a file named `httpsd.crt` in your `/etc/httpd/conf` directory. Follow the steps outlined in section 3.6 on page 58 to install your certificate.

3.7.2 Purchasing a Certificate From Thawte

To purchase a certificate from Thawte, follow these instructions:

1. Point your browser to `http://www.thawte.com/certs/server/request.html`, where Thawte provides an overview of the necessary steps.

2. The first thing you need to do is gather the documents that they will require, as discussed in both 3.3.2 on page 48 and the aforementioned web page.

3. The next step they describe is to generate a key and a certificate signing request (CSR). If you followed the instructions contained in section 3.4.1 on page 50 and 3.4.2 on page 51, you already have a key (`httpsd.key`) and a CSR (`httpsd.csr`). If you did not already create your key and certificate request, do so now.

4. Go to Thawte's "Buy an Organizational Certificate" web page at `https://www.thawte.com/cgi/server/step1.exe`. Select **Web Server Certificate**. Click on the **Next** button at the bottom of the page.

5. The next page, "Server Cert Enrollment," is shown as figure 3.10 on page 72. Paste the contents of your `httpsd.csr` file into the "Server Certificate Signing Request (CSR)" text box.

6. Choose **Apache+mod_ssl** from the "Web Server Software" pull-down menu.

7. Choose how you want to pay for the certificate.

8. Since Red Hat is a Hosting Partner with Thawte, you will receive $25 off the certificate price, so make sure you type in `redhat` to the "Hosting Partner Code" box.

9. Click on **Next** at the bottom of the page.

10. The next page, also entitled "Server Cert Enrollment" contains the form you need to fill out and is shown as figure 3.11 on page 73.

    Under "Background Information" select a description for your organization from the pull-down menu, or type your own description into the text box provided.

11. If you have a D-U-N-S number, type it into the text box under "DUNS Number."

12. Fill in contact information for the person in your organization who will be signing the authorization letter, as described in section 3.3.2 on page 48.

13. Under "Web Server Administrator," fill in contact information for your secure web server's administrator or your webmaster.

14. Click on the **Next** button at the bottom of the page.

15. The next page, also entitled "Server Cert Enrollment," is the last page of their enrollment form and is shown as figure 3.12 on page 74. From the first pull-down menu, choose the currency in which you are going to pay Thawte.

16. Type the street address for your organization into the "Office Street Address" text box.

17. Type in your organization's fax number into the text box under "Office Fax Number."

18. From the pull-down menu under "Nearest Thawte Office," choose the Thawte office closest to your organization.

19. If you want to, you may type a password or challenge phrase into the text box under "Privacy Protection Password." After you've submitted your application, you'll be able to check on its status on the web. If you type a password in here, then only people who know the password will be able to check on the status of your application. If you don't fill one in, anyone can check on your application.

20. Click on **Next** at the bottom of the page.

21. The next page will indicate that your submission is complete, as shown in figure 3.13 on page 75. This page provides you with a tracking number for your application, so that you can monitor its status over the web. This page also gives you contact information for Thawte and instructions on where to send your documentation (as described in section 3.3.2 on page 48) and your payment. You should print this page for future reference.

    After Thawte receives your documentation and payment, your certificate should be issued by email within five working days.

22. Save your certificate into a file named `httpsd.crt` in your `/etc/httpd/conf` directory. See section 3.6 on page 58 for instructions on installing your certificate.

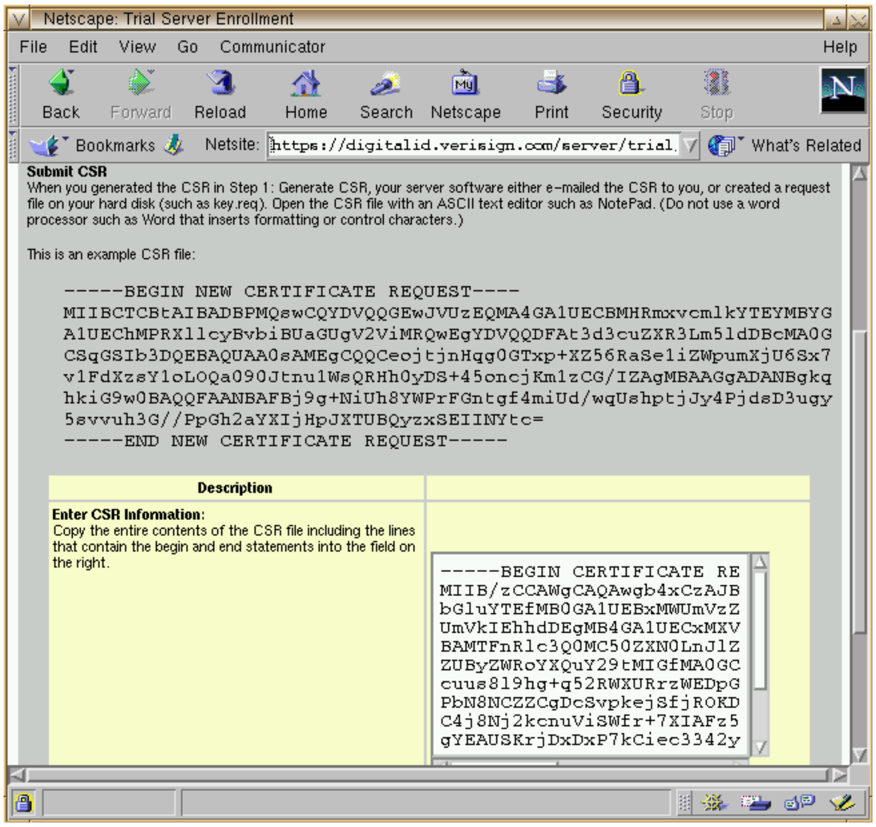

Figure 3.2: Paste in Your `httpsd.csr`

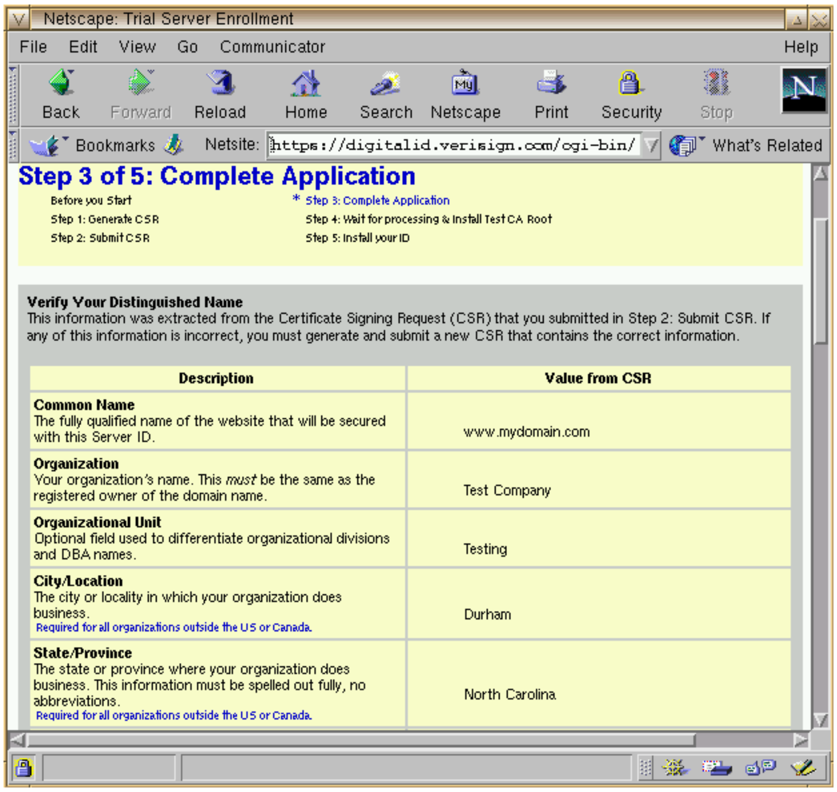

Figure 3.3: Application

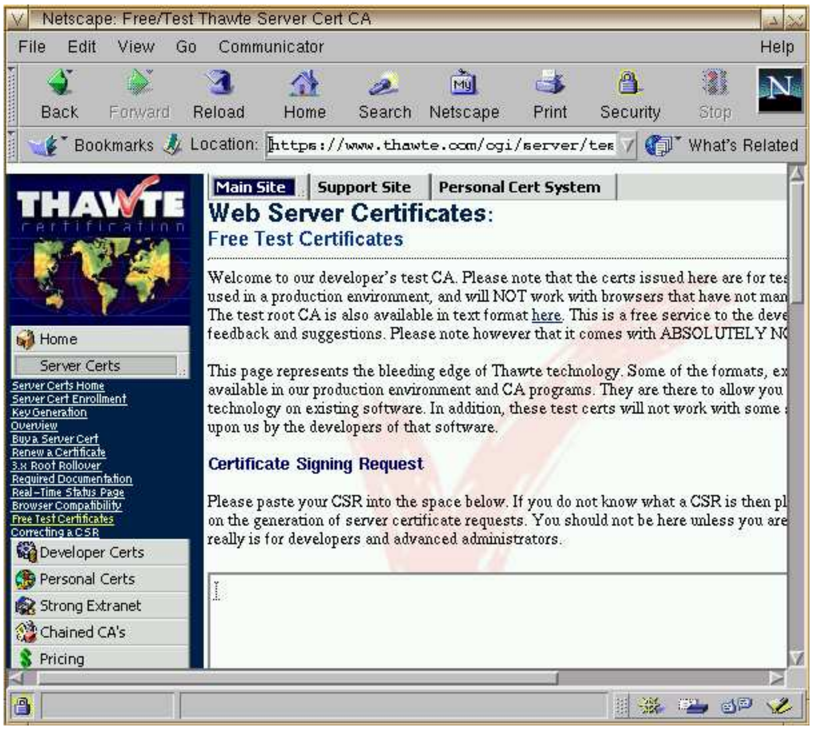

Figure 3.4: Thawte's Test Certificate Page

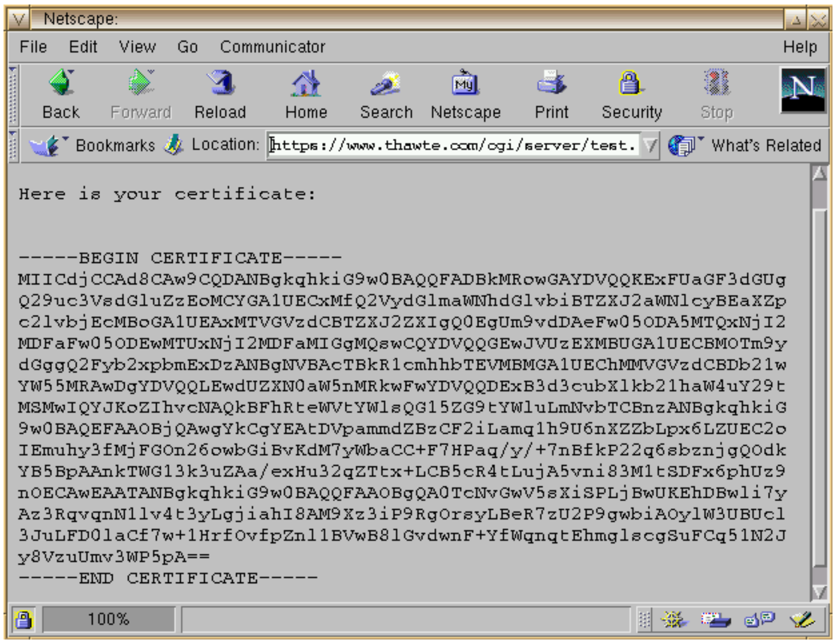

Figure 3.5: Thawte's Test Certificate Page

Figure 3.6: The Red Hat Secure Web Server Default Home Page

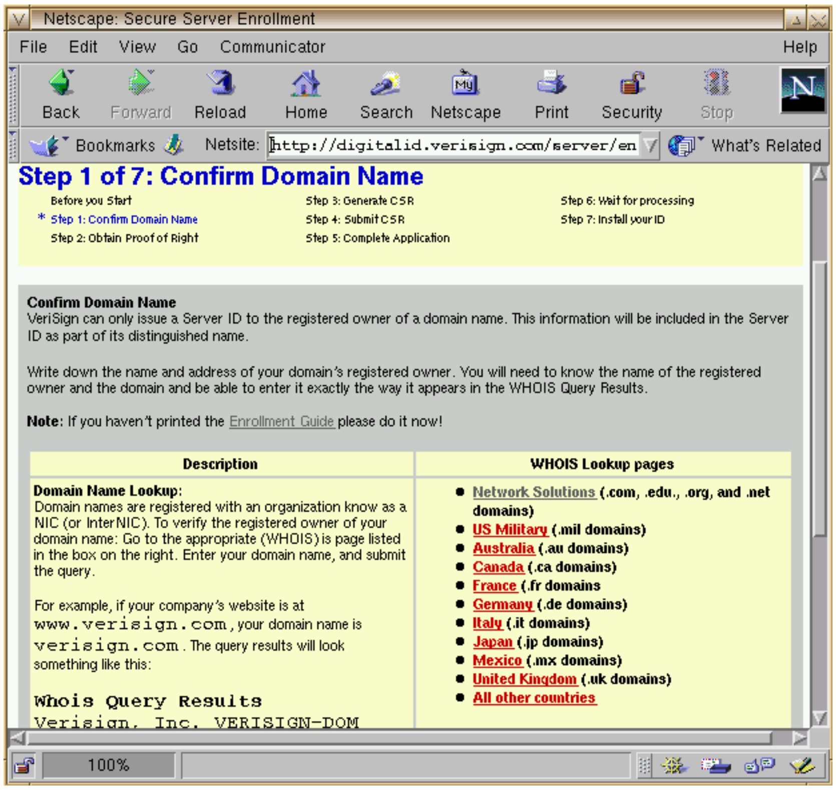

Figure 3.7: Confirming your Domain Name for VeriSign

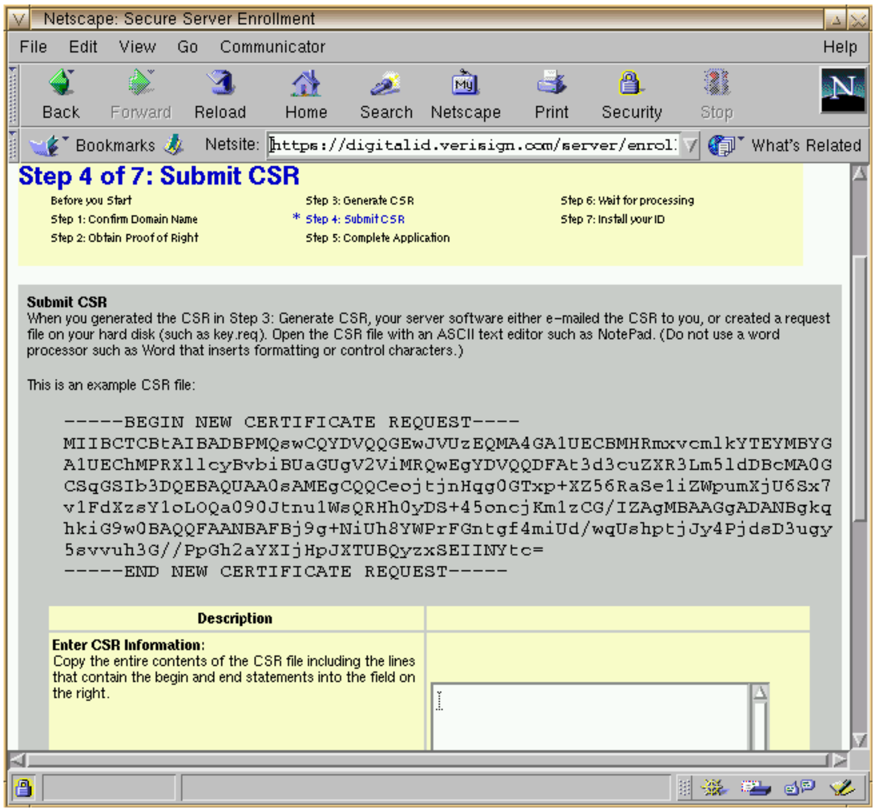

Figure 3.8: Submitting a CSR to VeriSign

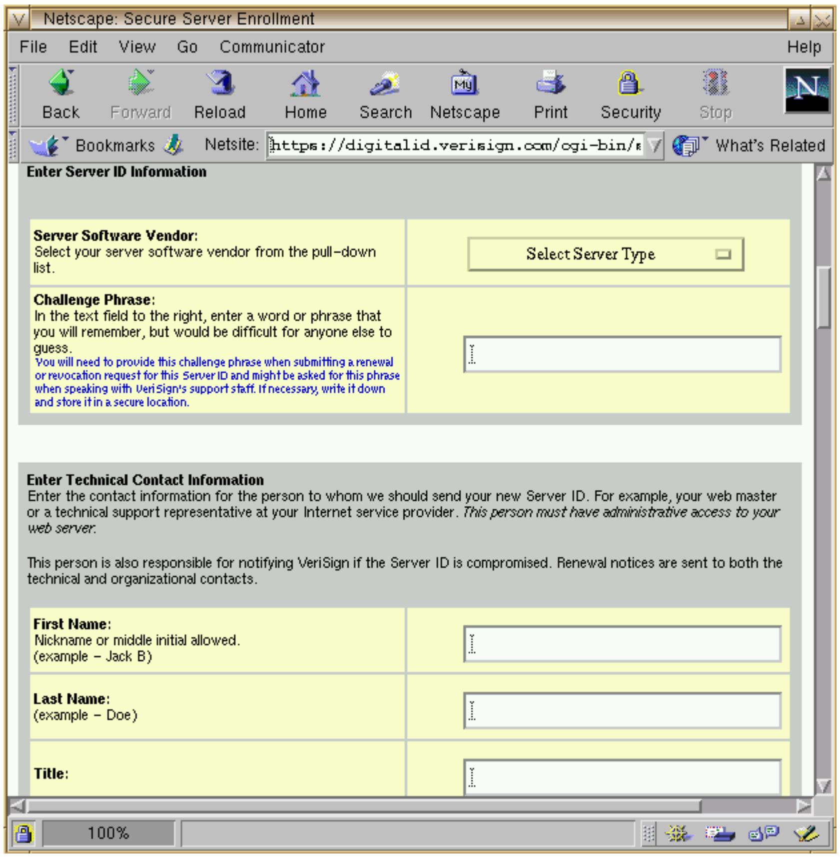

Figure 3.9: Completing the VeriSign Application

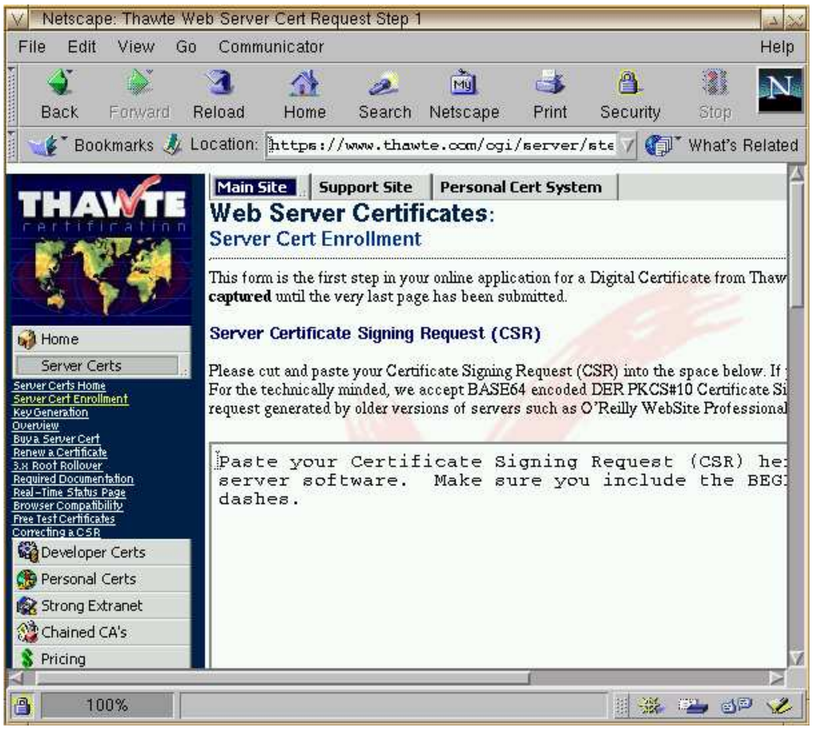

Figure 3.10: Thawte Enrollment Form

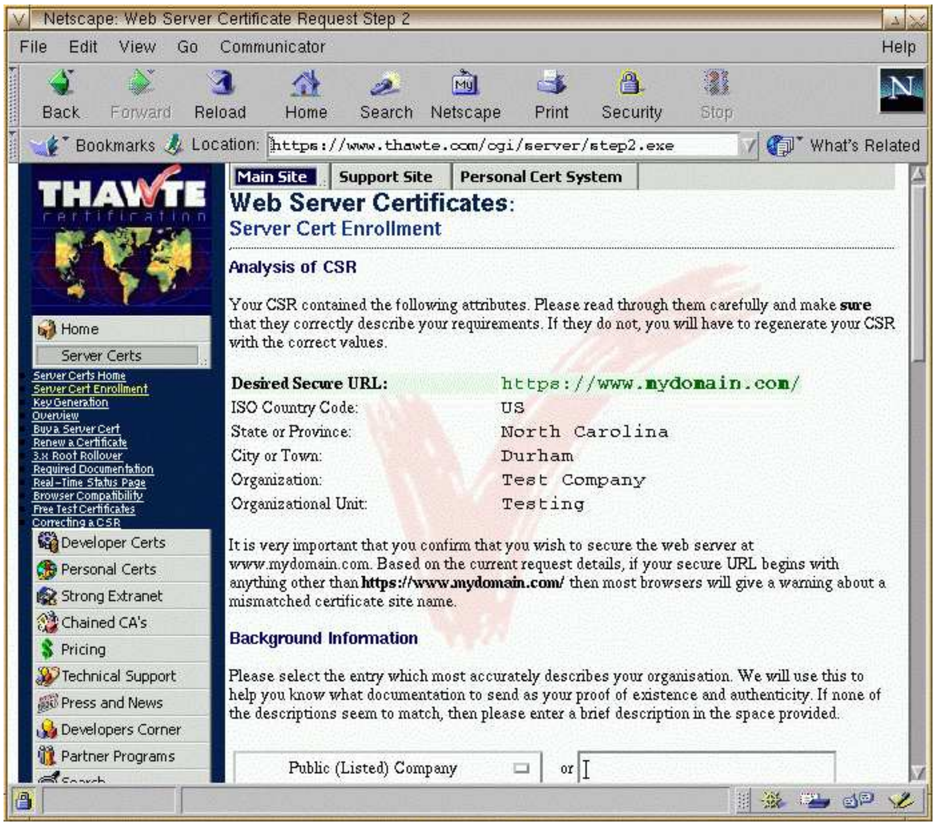

Figure 3.11: Thawte Application

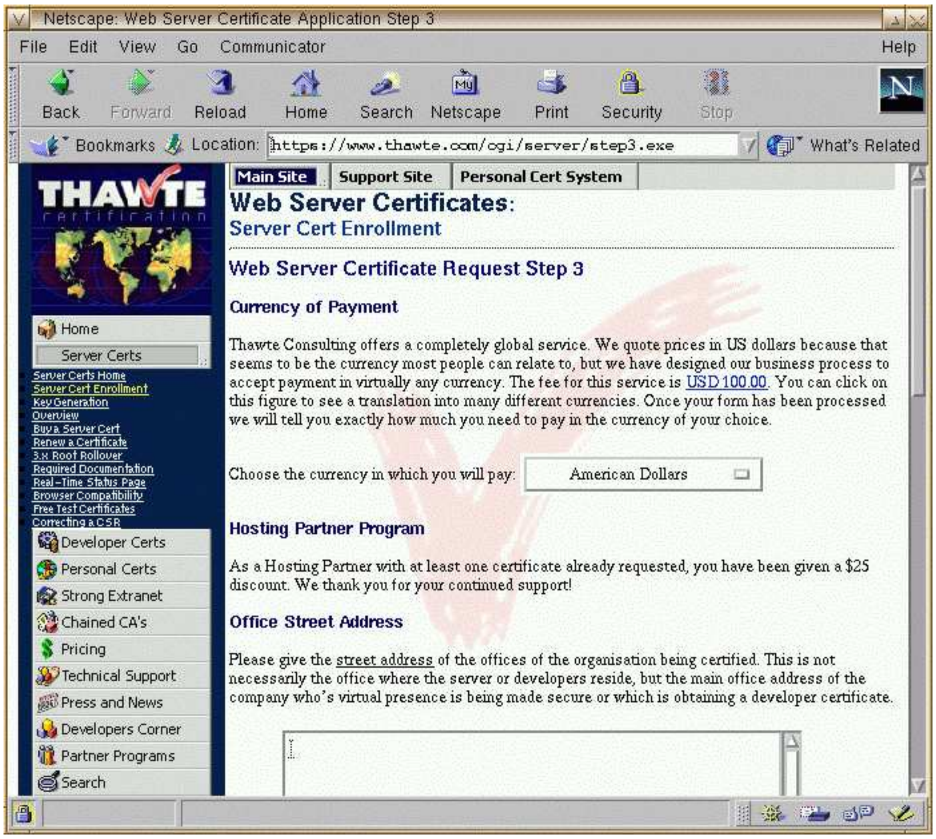

Figure 3.12: Thawte Application

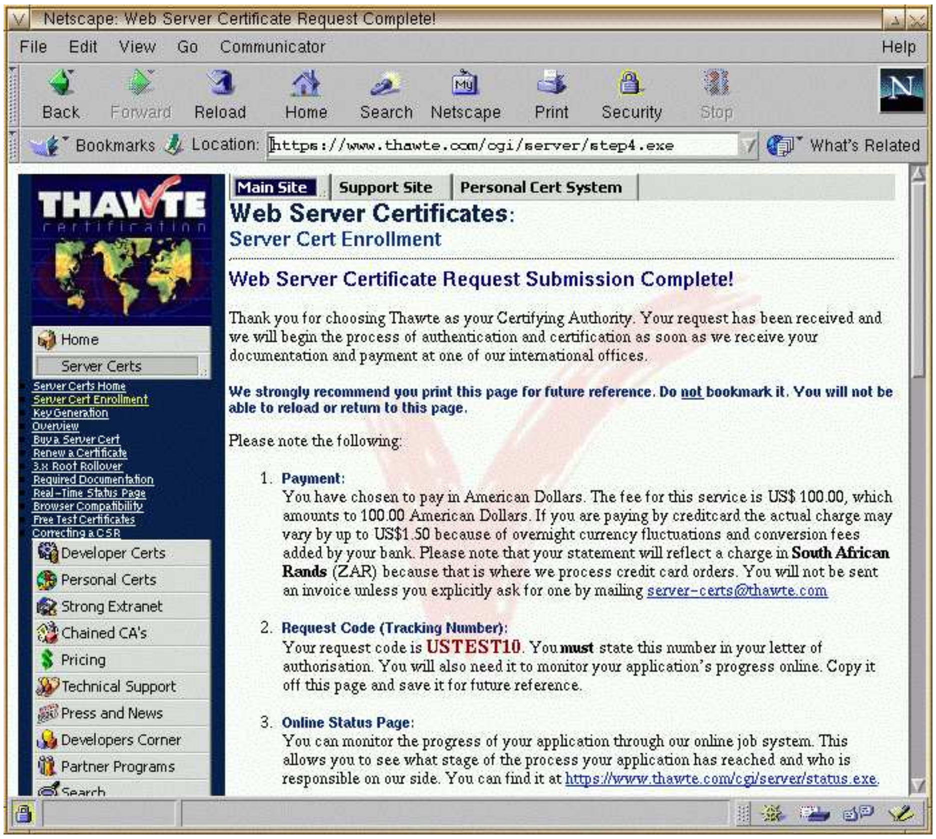

Figure 3.13: Thawte Submission Complete

4

Configuring Optional Packages

4.1 Configuring Analog

Analog is a complex program. Please refer to the Analog web page at `http://http://www.statslab.cam.ac.uk/~sret1/analog/` for information on how to set up and run Analog.

Basically, Analog parses your web server's log files and creates output based on that data. When configuring Analog, you need to tell it where to find your server's log files and what to do with the reports that it produces. To do this, edit the Analog configuration file, `/etc/analog.cfg`.

Change the LOGFILE and HOSTNAME lines to read as follows:

```
LOGFILE /var/log/httpd/access_log-ssl
HOSTNAME "Your Company's Name"
```

Then create an OUTFILE line and a HOSTURL line anywhere in the file like the following:

```
OUTFILE /path/to/analog/output.html
HOSTURL www.yourdomain.com
```

Obviously, you'll have to replace the OUTFILE path with a valid path to wherever you want Analog to leave its reports. Replace the www.yourdomain.com with your web server's domain name.

You'll probably want to put Analog's report file into a separate directory and restrict access to it using the access.conf file (or a .htaccess file). For many reasons, you may not want the world to be able to see your site's access log.

Analog can create many different types of reports. To turn on a daily report, add a line like the following to analog.cfg:

```
DAILY ON
```

For more information about the types of reports Analog can produce and how to configure the program in greater detail, please refer to the Analog web page at http://www.statslab.cam.ac.uk/~sret1/analog/.

4.2 Configuring mod_perl

To run mod_perl, you'll need to uncomment two lines in your httpd.conf file.

After the main LoadModule section, there's a list of three additional modules. Uncomment the perl_module line so that instead of:

```
#LoadModule perl_module modules/libperl.so
```

It reads:

```
LoadModule perl_module modules/libperl.so
```

After the main AddModule section, there's another list of three additional modules. Uncomment the libperl.c line so that instead of:

```
#AddModule libperl.c
```

It reads:

```
AddModule libperl.c
```

After you've uncommented these two lines, save the httpd.conf file and then restart your server as described in section 2.7 on page 40. You should now be running mod_perl.

The mod_perl package package is large and fairly complicated. It would be impossible to include all it can do in this guide. This section will explain how to set up mod_perl in its most commonly used role as a replacement for CGI.

The recommended way to do this is to edit the srm.conf file. Uncomment (remove the #PERL#) the following directives. Instead of:

```
#PERL#Alias /perl/ /home/httpd/perl/
#PERL#<Location /perl>
#PERL#SetHandler perl-script
#PERL#PerlHandler Apache::Registry
#PERL#Options +ExecCGI
#PERL#</Location>
```

The lines should read:

```
Alias /perl/ /home/httpd/perl/
<Location /perl>
SetHandler perl-script
PerlHandler Apache::Registry
Options +ExecCGI
</Location>
```

/home/httpd/perl/ can be any directory where you want to store your Perl scripts. The /home/httpd/perl/ directory will be like your cgi-bin directory. /perl/ is the directory where you'll access scripts on your server with URLs like http://www.domain.com/perl/.

If you want to have mod_perl handle scripts based just on file extensions, add these lines to your srm.conf file (but leave the previous lines commented out):

```
<Files *.pl>
SetHandler perl-script
PerlHandler Apache::Registry
Options ExecCGI
</Files>
```

With this configuration, mod_perl should handle any URLs like http://www.domain.com/yourscript.pl.

You'll need to restart your server for these modifications to take effect, as described in section 2.7 on page 40.

For more information on using mod_perl as a CGI replacement refer to the cgi_to_mod_perl Perl documentation by running the command:

```
perldoc cgi_to_mod_perl
```

For more information on mod_perl's configuration and capabilities, please refer to the Apache/Perl Integration Project web pages at http://perl.apache.org/.

4.3 Configuring mod_php

Like mod_perl, mod_php is a very large package. PHP is a complete scripting language, with various capabilities. To run PHP with your web server, you'll need to edit both the `httpd.conf` and the `srm.conf` files.

In your `httpd.conf` file, you need to uncomment the `LoadModule` and `AddModule` lines for the version or versions of PHP that you installed.

After the main `LoadModule` section, there's a list of three additional modules. If you installed the PHP/FI package (version 2.01), uncomment the `php_module` line so that instead of:

```
#LoadModule php_module  modules/mod_php.so
```

It reads:

```
LoadModule php_module  modules/mod_php.so
```

If you installed the PHP3 package, uncomment the following line instead:

```
#LoadModule php3_module modules/libphp3.so
```

If you installed both the PHP/FI package and the PHP3 package, uncomment both of the previous lines.

After the main `AddModule` section, there's another list of three additional modules.

If you installed the PHP/FI package, uncomment the `mod_php.c` line so that instead of:

```
#AddModule mod_php.c
```

It reads:

```
AddModule mod_php.c
```

If you installed the PHP3 package, uncomment the `mod_php3.c` line so that instead of:

```
#AddModule mod_php3.c
```

It reads:

```
AddModule mod_php3.c
```

If you installed both packages, uncomment both lines. Save your `httpd.conf` file.

Next, you will need to edit your `srm.conf` file.

If you installed the PHP/FI package, you will need to uncomment the `#AddType application/x-httpd-php .phtml` line. Instead of:

```
#AddType application/x-httpd-php .phtml
```

The line should read:

```
AddType application/x-httpd-php .phtml
```

If you installed the PHP3 package, you will need to uncomment two lines. Instead of:

```
#AddType application/x-httpd-php3 .php3
#AddType application/x-httpd-php3-source  .phps
```

The lines should read:

```
AddType application/x-httpd-php3 .php3
AddType application/x-httpd-php3-source .phps
```

If you installed both the PHP/FI and PHP3 packages, you'll need to uncomment all three of the previous `AddType` lines.

After you've uncommented the appropriate lines, save the srm.conf file and then restart your server as described in section 2.7 on page 40. You should now be running mod_php.

Please refer to the PHP web site at http://www.php.net/ for more information about configuring, testing and using PHP.

4.4 Configuring Apache-ASP

To enable Apache-ASP, you'll need to edit your access.conf file. Add the following lines to the end of the file:

```
<Location /asp/>
SetHandler perl-script
PerlHandler Apache::ASP
PerlSetVar Global /tmp
</Location>
```

Stop and start your web server, as described in Section 2.7. Apache-ASP will now serve pages out of the /asp directory.

Note that you will also need to have the mod_perl package installed in order to use Apache-ASP.

4.5 Configuring Squid

indexconfiguration!squid

To get started running the Squid caching proxy server, begin with the QUICKSTART file. QUICKSTART should be in /usr/doc/squid-1.1.22/, if you installed the squid package, or in /usr/doc/squid-novm-1.1.22/ if you installed squid-novm. For more complete information, you should open the FAQ.html file (in the same directory as the QUICKSTART file) using your web browser. You should also rely on the information provided at the Squid web site at http://squid.nlanr.net/.

You can have Squid start and stop automatically as your machine boots and shuts down by adding symlinks to your system's `/etc/rc.d/rc3.d` and `/etc/rc.d/rc0.d` directories. The following commands will create the symlinks:

```
cd /etc/rc.d/rc3.d
ln -s ../init.d/squid S99squid
cd /etc/rc.d/rc0.d
ln -s ../init.d/squid K99squid
```

If you are using the X Window System, you can create the symlinks with `tksysv`.

To run Squid, you'll need to edit some parameters in its configuration file, `/etc/squid.conf`, as follows:

1. Find the line that begins with:

   ```
   #cache_mem 8
   ```

   Uncomment this line and replace `8` with the amount of RAM (in megabytes) that you want to allocate to Squid.

2. Find the line that begins with:

   ```
   #cache_swap 100
   ```

   Uncomment this line and replace `100` with the amount of disk space (in megabytes) you want to allocate to Squid.

3. Find the following section:

   ```
   acl manager proto cache_object
   acl localhost src 127.0.0.1/255.255.255.255
   acl all src 0.0.0.0/0.0.0.0
   ```

   Add the following line to the end of the previous section:

   ```
   acl allowed_hosts src XXX.XXX.XXX.0/255.255.255.0
   ```

   Replace the `X`s (before the `/`) with your domain subnet. Replace the `255.255.255.0` with your domain's subnet mask.

4. Find the following section:

   ```
   http_access allow all
   ```

   Replace it with:

   ```
   http_access allow allowed_hosts
   http_access deny all
   ```

5. Find the following section:

   ```
   icp_access allow all
   ```

   Replace it with:

   ```
   icp_access allow allowed_hosts
   icp_access deny all
   ```

6. Find the following line:

   ```
   #cache_mgr webmaster
   ```

   Uncomment this line and replace `webmaster` with the email address of the Squid adminstrator.

7. Save the `squid.conf` file.

8. You can now start squid by invoking the `squid start` command in `/etc/rc.d/init.d/`:

   ```
   /etc/rc.d/init.d/squid start
   ```

   If this is the first time you've run Squid, it will create the swap directories it needs upon startup.

Squid should now be running on port 3128 of your web server. You can now set up Netscape (from any machine you chose to allow access to Squid) to use your machine as its proxy server. Some people configure their routers to forward any requests on port 80 to redirect through Squid; this saves them from having to set up the proxy on all the machines on their network.

4.6 Configuring ht://Dig

For complete documentation on configuring and running ht://Dig, point to the file `/usr/doc/htdig-3.0.8b2/index.html` with your web browser or refer to the ht://Dig web page at `http://www.htdig.org`.

Please Note:ht://Dig does not currently have the ability to connect to a secure web server. If you want to use ht://Dig, you'll have to use it with the non-secure web server which was provided as part of your secure web server's default configuration.

To get started with ht://Dig, edit the `/etc/htdig.conf` file as follows:

1. Change the line that begins with:

   ```
   start_url: http://htdig.sdsu.edu/
   ```

   so that it names your domain instead of `htdig.sdsu.edu`.

2. Save the `htdig.conf` file. You may wish to read through this file and make any other changes you think are appropriate, but this is the only required change.

3. Type the following commands to set up your search database:

   ```
   rundig
   ```

   `rundig` is in `/usr/bin`, which should be in your path.

4. Now point your browser to `http://www.yourdomain.com/htdig/search.html` to search your web site.

5. You may also want to customize the following files to display information specific to your site:

   /home/httpd/htdig/search.html This is the page that you will need to link to for searching your site. You may wish to make this file a symlink to `index.html` in the same directory, so that the `http://www.yourdomain.com/htdig/` works as well.

   /var/lib/htdig/header.html This page will be displayed at the top of any search results.

/var/lib/htdig/footer.html This page will be displayed at the bottom of any search results.

/var/lib/htdig/nomatch.html This page will be displayed when no search results have been found.

/var/lib/htdig/syntax.html This page will be displayed when a syntax error has occurred in a boolean expression.

6. If your site changes often, you'll need to set up `htdig` and `htmerge` to run automatically using `cron`. See your system documentation for more information on `cron`.

Index

A

access.conf file 32
acknowledgements ix
analog 4
    configuration 77
Apache
    configuration 16
    installing 1
    reload 40
    restart 40
    running without security 36
    securing 43
    starting 40
    stopping 40
Apache perl module 4
Apache PHP/FI module 5
Apache security
    explanation of 44
Apache-ASP
    configuration 83
    package 6

B

buying a certificate 59

C

CA . . . . . *see* certificate authorities
caching proxy server . . . *see* squid
CD-ROM
    mounting 3
certificate
    buying 59
    buying from Thawte 61
    buying from VeriSign 59
    creation of request 49, 51
    documents required 46
    installing 58
    purchasing 59
    purchasing from Thawte . 61
    purchasing from VeriSign 44, 59
    test, creating 54
    test, obtaining 54
    testing 58
certificate authorities
    choosing 46
certificate request
    creation of 51
choosing a CA 46
Communicator, Netscape 9
configuration
    analog 77
    Apache 16
    Apache-ASP 83
    ht://Dig 86
    mod_perl 78
    mod_php 81
    optional packages 77
    PHP/FI 81

- server 15
- SSL 25
- virtual hosts 36

copyright ii
creating certificate request 49, 51
creating key 49, 50

D

D-U-N-S numbers 46
devel package 6
documentation, how to improve vi
DocumentRoot, changing 36
documents required for certification 46
dynamically shared objects . . . 34

F

feedback, how to give vi
files
- access.conf 32
- httpd.conf 17
- srm.conf 27

FrontPage 15

H

ht://Dig 8
- configuration 86

httpd.conf file 17

I

installation
- starting 9

installer
- starting 9

installing Apache 1
installing certificates 58
introduction v

K

key, creating 49, 50

L

log parsing program . . *see* analog

M

mod_perl 4
- configuration 78

mod_php
- configuration 81

modules
- compiling in 34
- loading 34
- Perl Apache 4
- PHP/FI 5
- PHP3 5

mounting CD-ROM 3

N

Navigator, Netscape 9
Netscape
- Communicator 9
- Navigator 9

non-secure server, disabling . . 36
numbers, D-U-N-S 46

O

objects, dynamically shared . . 34
operating system version 2
optional packages 3, 77

P

packages
- optional 3

packages, optional 77
perl, Apache 4

PHP/FI
    configuration 81
PHP/FI, Apache 5
PHP3 5
port numbers 42
purchasing a certificate 59

S

search engines . . . . . . *see* ht://Dig
securing
    Apache 43
    server 43
security
    configuring 25
    explanation of 44
    running Apache without . 36
server
    accessing 42
    configuration 15
    reload 40
    restart 40
    securing 43
    starting 40
    stopping 40
server security
    explanation of 44
software version 2
source package 7
squid 7
    configuration 83
squid-novm 8
srm.conf file 27
SSL directives 25
starting
    Apache 40
starting installation 9
starting server 40
stopping
    Apache 40
stopping server 40

T

test certificate
    creating 54
    obtaining 54
    obtaining from Thawte . . 57
    obtaining from VeriSign . . 55
testing certificates 58
Thawte
    buying a certificate 61
    obtaining test certificate from 57
    proving identity to 48
    purchasing a certificate . . 61
    test certificate 57
trademark statements ii

U

URLs 42

V

VeriSign
    buying a certificate 59
    obtaining test certificate from 55
    proving identity to 47
    purchasing a certificate . . 59
    test certificate 55
versions
    operating system 2
    software 2
virtual hosts, configuring 36

W

web browser . . . . *see* Netscape