Chapter 9:
Configuring a basic Apache server

Learning objectives

Upon completing this chapter, the learner should be able to:

- Understand Apache web server basics
- Configure a basic Apache web server
- Manage virtual hosts
- Manage TLS settings

Key terms

Apache server

Apache configuration file

web server

web server content

httpd

httpd.conf

Apache virtual hosts

chroot

ssl.conf

TLS

public key

private key

TLS certifiate

virtual host

Table of content

1. Working with Apache server 3
   1.1. Installing the required software 4
   1.2. Identifying the main configuration file 7
   1.3. Creating web server content 12
   1.4. Understanding Apache configuration files 14
   1.5. Creating Apache virtual hosts 17
2. Configuring Apache for using TLS 20
   2.1. Creating a self signed certificate 20
   2.2. Setting up a virtual host 24

1. Working with Apache server

Configuring a basic Apache server is not hard to do, it consists of a few easy steps:

- Install the required software
- Identify the main configuration file
- Create some web server content

1.1. Installing the required software

On RHEL systems, the Apache server is provided through some different software packages. The basic package is httpd, this package contains everything that is needed for an operational but basic web server. There are some additional packages, as well. For a complete overview, you can use the yum search http command and use yum install httpd to install the base package.

```
[root@server1:~]# yum search http
...
http-parser.i686 : HTTP request/response parser for C
http-parser.x86_64 : HTTP request/response parser for C
...
httpd.x86_64 : Apache HTTP Server
httpd-devel.x86_64 : Development interfaces for the
Apache HTTP server
httpd-manual.noarch : Documentation for the Apache HTTP
server
httpd-tools.x86_64 : Tools for use with the Apache HTTP
Server
...
perl-HTTP-Tiny.noarch : Small, simple, correct HTTP/1.1
client
...
xmlrpc-c.x86_64 : A lightweight RPC library based on XML
and HTTP

  Name and summary matches only, use "search all" for
everything.
```

Notice that the yum search http command gives a lot of packages. This is because the Apache web server is modular and the different modules are provided through additional yum packages.

Instead of using the individual software packages, you can also use yum groups. The yum groups list command gives an overview of all yum groups that are available, and the Basic Web Server yum group provides all you need to install the Apache web server and its core requirements. Use yum groups install "Basic Web Server" to install it.

```
[root@server1:~]# yum groups list
...
Available Environment Groups:
   Minimal Install
   ...
   Basic Web Server
   Virtualization Host
   Server with GUI
   GNOME Desktop
   KDE Plasma Workspaces
   Development and Creative Workstation
Available Groups:
   ...
   System Administration Tools
   System Management
Done
```

To verify the installation and availability of the Apache web server you can use the systemctl status command as shown below:

```
[root@server2 ~]# systemctl status httpd
httpd.service - The Apache HTTP Server
Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled)
Active: active (running) since Sat 2015-05-16 04:27:49 PDT; 1s ago
Main PID: 42997 (httpd)
Status: "Processing requests..."
CGroup: /system.slice/httpd.service
├─42997 /usr/sbin/httpd -DFOREGROUND
├─42998 /usr/sbin/httpd -DFOREGROUND
├─42999 /usr/sbin/httpd -DFOREGROUND
├─43000 /usr/sbin/httpd -DFOREGROUND
├─43001 /usr/sbin/httpd -DFOREGROUND
└─43002 /usr/sbin/httpd -DFOREGROUND
May 16 04:27:49 server2.example.com systemd[1]: Started The Apache HTTP Server.
```

1.2. Identifying the main configuration file

The configuration of the Apache web server goes through different configuration files. The section "Understanding Apache configuration files" later in this chapter provides an overview of the way these files are organized.

The main Apache configuration file is /etc/httpd/conf/httpd.conf, in this section, many parameters are specified.

- The most important parameter to understand for setting up a basic web server is the **DocumentRoot** parameter, which specifies the default location where the Apache web server looks for its contents.
- Another important configuration parameter is the **ServerRoot**. This defines the default directory where Apache will look for its configuration files. By default, the /etc/httpd directory is used for this purpose, but alternative directories can be used as well. You notice that in the httpd. conf many other configuration files are referred to.

The names of these configuration files are all relative to the ServerRoot /etc/httpd, the listing below shows partial contents of the /etc/httpd/conf/httpd.conf configuration file:

```
[root@server1 ~]# cat /etc/httpd/conf/httpd.conf | grep -
v '#'
ServerRoot "/etc/httpd"
Listen 80
Include conf.modules.d/*.conf
User apache
Group apache
ServerAdmin root@localhost
<Directory />
     AllowOverride none
     Require all denied
</Directory>
DocumentRoot "/web"
<Directory "/var/www">
     AllowOverride None
     Require all granted
</Directory>
<Directory "/web">
     Options Indexes FollowSymLinks
     AllowOverride None
     Require all granted
</Directory>
<IfModule dir_module>
     DirectoryIndex index.html
```

```
</IfModule>
<Files ".ht*">
     Require all denied
</Files>
ErrorLog "logs/error_log"
LogLevel warn
<IfModule log_config_module>
     LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\"
\"%{User-Agent}i\"" combined
     LogFormat "%h %l %u %t \"%r\" %>s %b" common
<IfModule logio_module>
     LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\"
\"%{User-Agent}i\" %I %O" combinedio
</IfModule>
CustomLog "logs/access_log" combined
</IfModule>
<IfModule alias_module>
     ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"
</IfModule>
```

```
<Directory "/var/www/cgi-bin">
     AllowOverride None
     Options None
     Require all granted
</Directory>
<IfModule mime_module>
     TypesConfig /etc/mime.types
     AddType application/x-compress .Z
     AddType application/x-gzip .gz .tgz
     AddType text/html .shtml
     AddOutputFilter INCLUDES .shtml
</IfModule>
AddDefaultCharset UTF-8
<IfModule mime_magic_module>
     MIMEMagicFile conf/magic
</IfModule>
EnableSendfile on
IncludeOptional conf.d/*.conf
```

The table below shows essential configuration parameters overview from the above file:

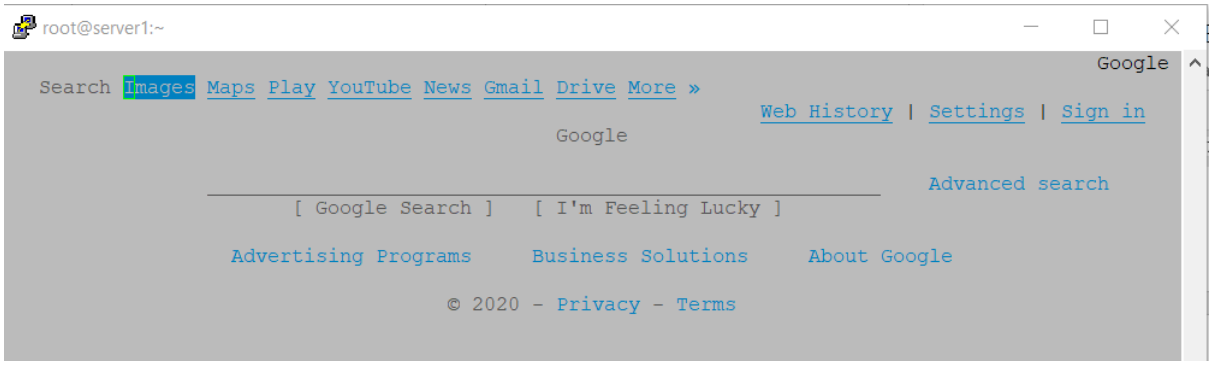
Parameter	Explanation
ServerRoot	The directory that contains all server configuration. Names of other configuration files are relative to this directory.
Listen	The port that the httpd process listens on.
Include	Used to refer to directories that contain additional configuration files that need to be included.
ServerAdmin	The name of the server administrator.
Directory	Used as a block of parameters to specify parameters that are specific for one directory. Often used to determine which kind of content is allowed. A directory block often contains AllowOverride, Require, and Options as common directives.
AllowOverride	If set to None, httpd will not read the contents of the .htaccess file that can be used for per-directory settings
ErrorLog	Used to name the file where errors are logged to.

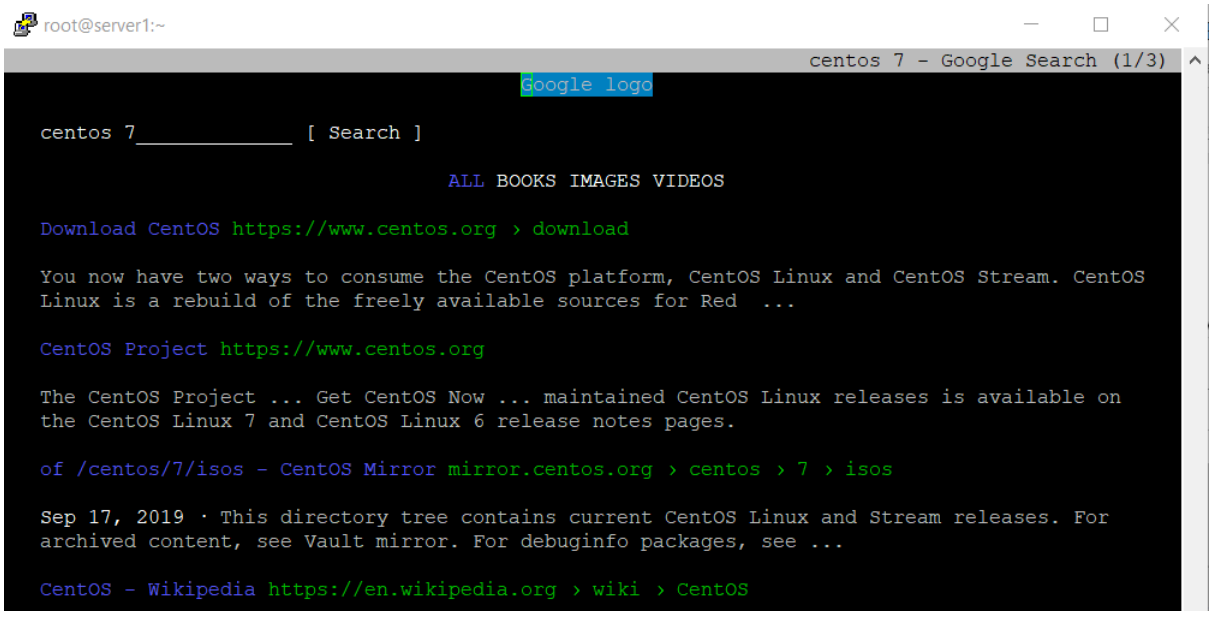
1.3. Creating web server content

After identifying the web server DocumentRoot, the Apache web server by default looks for a file with the name index.html and will present the contents of that document to clients using a browser to access the web server. It suffices to configure this file with very basic contents; just a line like “Welcome to my web server” will do.

To test the web server, you can launch a browser. The Firefox browser is installed by default on all graphical installations of RHEL7. If your server does not run a graphical interface, use yum install elinks to install the text-based elinks browser, this browser does not allow you to load complicated web pages, but it does allow you to verify the working of the web server.

The figures below show using the elinks to open Google site and then search for the term “centos7” in it:





Exercise: Setting up a basic web server

In this exercise, you learn how to set up a basic Apache web server. Nothing fancy, just enough to get you going and test web server functionality.

1. Type yum groups install “Basic Web Server”. This installs the httpd package, and some of the most commonly used additional packages as well.
2. Open the main Apache configuration file with an editor, and look up the line that starts with DocumentRoot. This identifies the location where the Apache server will look for the contents it will service. Confirm that it is set to /var/www/html.
3. In the directory /var/www/html, create a file with the name index.html. In this file, type “Welcome to my web server”.
4. To start and enable the web server, type systemctl start httpd; systemctl enable httpd. This starts the web server and makes sure that it starts automatically after restarting the server. Use systemctl status httpd to check that the web server is up and running.
5. Type yum install elinks to install the elinks text-based browser. Type elinks http://localhost to connect to the web server and verify it is working.

1.4. Understanding Apache configuration files

A default installation of the Apache web server creates a relatively complex configuration tree in the /etc/httpd directory. Listing shows the default contents of this directory. Notice that the contents of this directory may differ on your server if additional software has been installed. Apache is modular, and upon installation of additional Apache modules, different configuration files might be installed here.

```
[root@server1:~]# tree /etc/httpd/
/etc/httpd/
├── conf
│   ├── httpd.conf
│   └── magic
├── conf.d
│   ├── autoindex.conf
│   ├── fcgid.conf
│   ├── manual.conf
│   ├── README
│   ├── ssl.conf
│   ├── userdir.conf
│   └── welcome.conf
├── conf.modules.d
│   ├── 00-base.conf
│   ├── 00-dav.conf
│   ├── 00-lua.conf
│   ├── 00-mpm.conf
│   ├── 00-proxy.conf
│   ├── 00-ssl.conf
│   ├── 00-systemd.conf
│   ├── 01-cgi.conf
│   └── 10-fcgid.conf
├── logs -> ../../var/log/httpd
├── modules -> ../../usr/lib64/httpd/modules
└── run -> /run/httpd
6 directories, 18 files
```

The first thing you notice is the presence of three symbolic links to logs, modules, and a run directory. These are created to allow Apache to be started in a chroot environment, which in turn provides a fake root directory, which is a directory in the file system that is presented as the root directory for the process that is running in the chroot environment.

This is done for security reasons: Processes that are running in a chroot environment can access files in that chroot environment only, which decreases the risk of security incidents to happen when intruders manage to get a login shell using the web server identity and try walking through the file system to do unauthorized things.

The main configuration files for the Apache web server is in the /etc/httpd/conf directory.

- To start, there is the httpd.conf file, which contains the most important configuration parameters
- Apart from that, there is a file with the name magic. This file is used by the browser to interpret how the contents of the web server should be interpreted. It makes sure that the web server content is shown correctly in different browsers

The /etc/httpd/conf.d directory contains files that are included in the Apache configuration. This is done by the line Include conf.modules.d/*.conf in the httpd.conf file. As is the case for the ServerRoot, this approach makes it possible to add configuration files that define the different web pages without changing the contents of the /etc/httpd/conf/httpd. conf file

- The last configuration directory is /etc/httpd/conf.modules.d. Apache is a modular web server. Therefore, the functionality of the Apache web server can easily be extended by adding additional modules that enable many different features. If modules are used, they can use their own module specific configuration files, which will be dropped in the /etc/httpd/conf.modules.d directory

Again, the purpose of this approach is to keep the configuration in /etc/httpd/conf.d/httpd.conf as clean as possible and to make sure that module specific configuration is not overwritten if the Apache generic configuration is updated.

1.5. Creating Apache virtual hosts

Many companies host more than one website. Fortunately, it is not necessary to install a new Apache server for every website that you want to run. Apache can be configured to work with virtual hosts, which is a distinguished Apache configuration file that is created for a unique hostname. When working with virtual hosts, the procedure to access the host is roughly like the following:

1. The client starts a session to a specific virtual host, normally by starting a browser and entering the URL to the website the client wants to use.
2. DNS helps resolving the IP address of the virtual host, which is the IP address of the Apache server that can host different virtual hosts.
3. The Apache process receives requests for all the virtual hosts it is hosting.
4. The Apache process reads the HTTP header to analyze which virtual host this request needs to be forwarded to.
5. Apache reads the specific virtual host configuration file to find which document root is used by this specific virtual host.
6. The request is forwarded to the appropriate contents file in that specific document root.

When working with virtual hosts, there are a few things to be aware of:

- If your Apache server is configured for virtual hosts, all servers it is hosting should be handled by virtual hosts. To create and catch all entry for all HTTP requests that are directed to this host but that do not have a specific virtual host file, you can create a virtual host for _default_:80.
- Name-based virtual hosting is the most common solution. In this solution, virtual hosts are using different names but the same IP address.
- IP-based virtual hosts are less common, but are required if the name of a web server must be resolved to a unique IP address. IP-based virtual hosts do require several IP addresses on the same machine and are common in configuration where the Apache server uses TLS to secure connections.

Exercise: Installing Apache virtual hosts

In this exercise, you create two virtual hosts. To help you setting up virtual hosts, you first set up name resolution, after which you create the virtual hosts configuration as well.

1. On both server1 and server2, open the file /etc/hosts with an editor and add two lines that make it possible to resolve the names of the virtual host you are going to create to the IP address of the virtual machine:
   <server1-ip-address> account.example.com account
   <server1-ip-address> sales.example.com sales
2. On server1, open a root shell and add the following to the /etc/httpd/conf/httpd.conf file. (You can leave all other settings as they are.)
   <Directory /www/docs>
   Require all granted
   AllowOverride None
   </Directory>
3. On server1, open a root shell and create a configuration file with the name account.example.com.conf in the directory /etc/httpd/conf.d. Give this file the following content:
   <VirtualHost *:80>
   ServerAdmin webmaster@account.example.com

DocumentRoot /www/docs/account.example.com

ServerName account.example.com

ErrorLog logs/account/example.com–error_log

CustomLog logs/account.example.com–access_log common

</VirtualHost>

4. Close the configuration file and from the root shell use mkdir –p /www/docs/account.example.com.
5. Create a file with the name index.html in the account document root, and make sure its contents read “Welcome to account”.
6. Temporarily switch off SELinux using setenforce 0.
7. Use systemctl restart httpd to restart the Apache web server, make sure the directory /etc/httpd/logs/account/ is existing.
8. Use elinks http://account.example.com. You should now see the account welcome page. (You may have to install elinks, using yum install –y elinks.)
9. Back on the root shell, copy the /etc/httpd/conf.d/account.example.com.conf file to a file with the name /etc/httpd/conf.d/sales.example.com.conf.
10. Open the sales.example.com.conf file in vi, and use the vi command: %s/account/sales/g. This should replace all instances of account with the text sales.
11. Create the /www/docs/sales.example.com document root, and create a file index.html in it, containing the text “Welcome to the sales server”.
12. Restart httpd and verify that the account and the sales servers are both accessible from both server1 and server2.

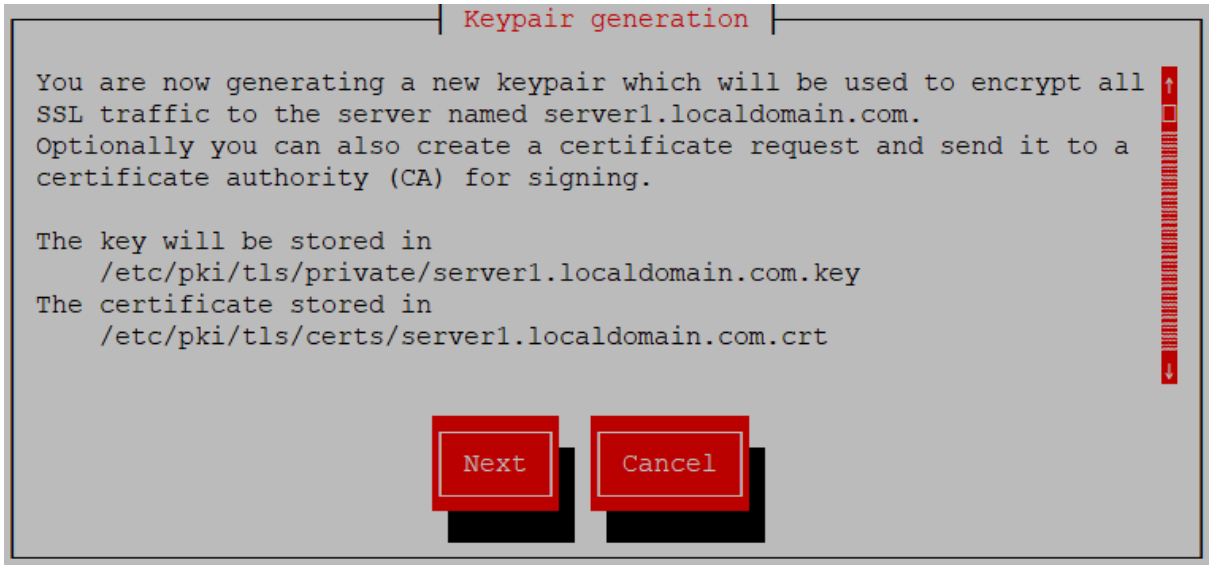
2. Configuring Apache for using TLS

By default, the identity of a web server is not verified. This opens your web server for man–in–the–middle attacks, where someone else is assuming your web server's identity. If additional security is required, the server can be configured with Transport Layer Security (TLS). In this section, you learn how to do this. When secured with TLS, the web server is configured with public/private key certificates to guarantee the identity of the web server. Using these keys makes it possible to verify the server identity but also to send data that is encrypted and therefore not readable while in transit.

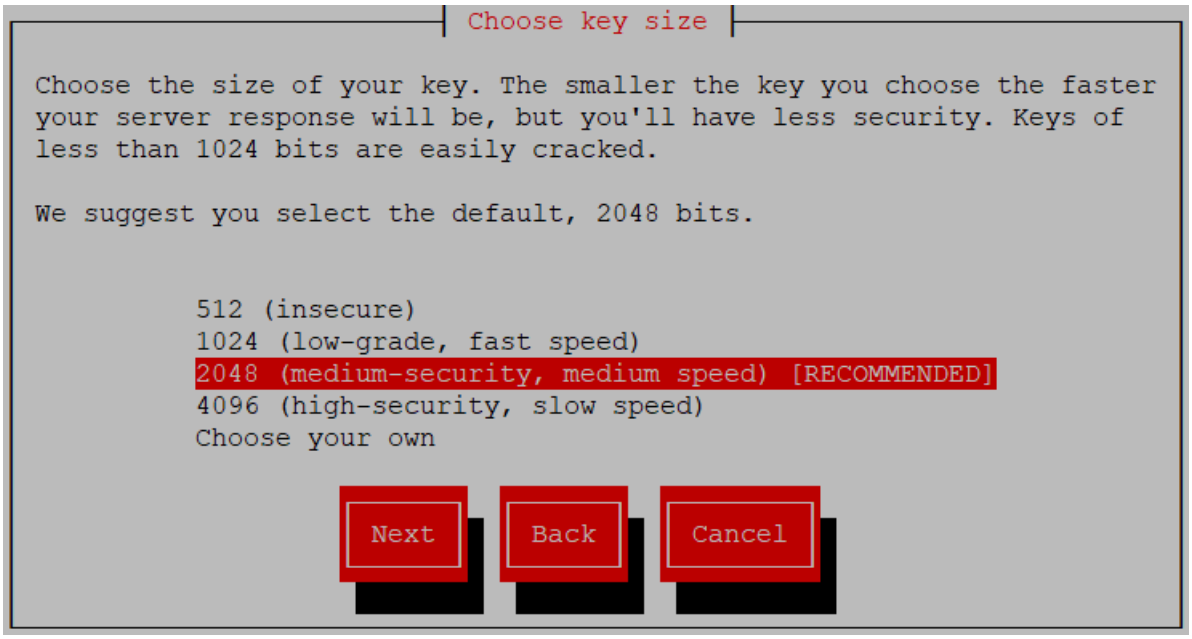
2.1. Creating a self signed certificate

The following procedure describes how these steps are applied to create a certificate for the sales.example.com server:

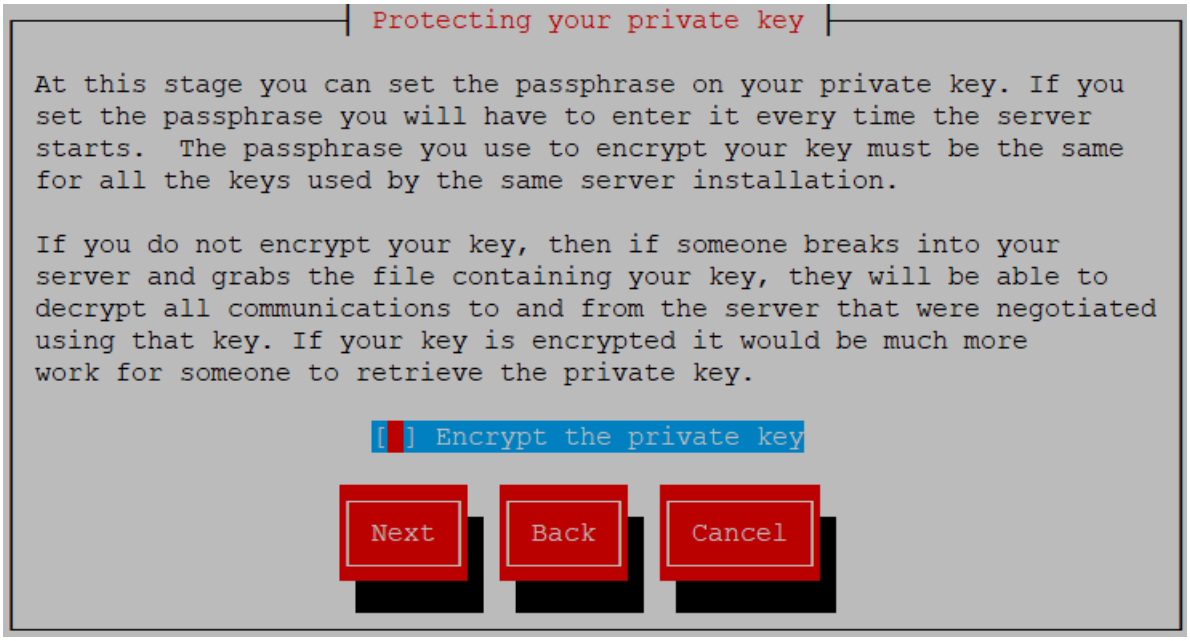
1. Type yum install crypto-utils mod_ssl to install the required packages.
2. Type genkey server1.localdomain.com. This opens the genkey utility, which tells you where the key and the certificate will be stored. From the first screen, select Next.



3. In the second step, the size of the key needs to be specified. A bigger size is higher security but slower. It is a good idea to accept the default of 2048 bits.



4. After selecting the key size, some random bits are generated. This takes some time. You can decrease the time it takes by performing some random activity on the server you are working on (like moving the mouse) or run the command `rngd -r /dev/urandom` which creates the entropy for you.



5. In this step, you need to provide the certificate details, specially the FQDN as shown below:

6. Finally, the command will confirm the operation and displays the location of the certificate and the keys we'd created, by default they are stored in the /etc/pki/tls/certs/ and /etc/pki/tls/private/ directory respectively.

```
[root@server1:~]# genkey server1.localdomain.com
/usr/bin/keyutil -c makecert -g 2048 -s
"CN=server1.localdomain.com, O=My Company Ltd, L=Newbury,
ST=Berkshire, C=GB" -v 1 -a -z /etc/pki/tls/.rand.14820 -
o /etc/pki/tls/certs/server1.localdomain.com.crt -k
/etc/pki/tls/private/server1.localdomain.com.key
cmdstr: makecert
cmd_CreateNewCert
command:  makecert
keysize = 2048 bits
subject = CN=server1.localdomain.com, O=My Company Ltd,
L=Newbury, ST=Berkshire, C=GB
valid for 1 months
random seed from /etc/pki/tls/.rand.14820
output will be written to
/etc/pki/tls/certs/server1.localdomain.com.crt
output key written to
/etc/pki/tls/private/server1.localdomain.com.key
Generating key. This may take a few moments...
Made a key
Opened tmprequest for writing
/usr/bin/keyutil Copying the cert pointer
Created a certificate
Wrote 1682 bytes of encoded data to
/etc/pki/tls/private/server1.localdomain.com.key
Wrote the key to:
/etc/pki/tls/private/server1.localdomain.com.key
```

2.2. Setting up a virtual host

To configure Apache for using TLS certificates, three steps must be accomplished:

- A certificate must be obtained, as described above
- The required Apache TLS modules must be installed
- The Apache (virtual) host must be configured to use the certificates

After generating the certificate and key, you can configure a virtual host with TLS. Configuring a virtual host with TLS is not much different from configuring a regular virtual host, you just have some additional parameters to deal with. A good starting point is the file /etc/httpd/conf. d/ssl.conf.

```
[root@server2 /]# grep -v '^#' /etc/httpd/conf.d/ssl.conf
Listen 443 https
SSLPassPhraseDialog      exec:/usr/libexec/httpd-ssl-pass-
dialog
SSLSessionCache shmcb:/run/httpd/sslcache(512000)
SSLSessionCacheTimeout 300
SSLRandomSeed startup file:/dev/urandom 256
SSLRandomSeed connect builtin
SSLCryptoDevice builtin
<VirtualHost _default_:443>
     ErrorLog logs/ssl_error_log
     TransferLog logs/ssl_access_log
     LogLevel warn
     SSLEngine on
     SSLProtocol all -SSLv2
     SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5
     SSLCertificateFile /etc/pki/tls/certs/localhost.crt
     SSLCertificateKeyFile
     /etc/pki/tls/private/localhost.key
     <Files ~ "\.(cgi|shtml|phtml|php3?)$">
     SSLOptions +StdEnvVars
     </Files>
     <Directory "/var/www/cgi-bin">
     SSLOptions +StdEnvVars
     </Directory>
BrowserMatch "MSIE [2-5]" \
nokeepalive ssl-unclean-shutdown \
downgrade-1.0 force-response-1.0
CustomLog logs/ssl_request_log \
"%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
</VirtualHost>
```

To configure a virtual host, start by changing the VirtualHost _default_:443 line to VirtualHost *:443. Then, change the ServerName to match the name of the server you are going to be used.

After changing the ServerName, you need to change the SSLCertificateFile and the SSLCertificateKeyFile to match the names of the files that you have just created.

If you want all traffic that comes in on the regular HTTP port 80 to be redirected to the TLS secured host, include the following block in the definition of the /etc/httpd/conf.d/sales.example.com.conf file you have created previously:

```
<VirtualHost *:80>
ServerName sales.example.com
RewriteEngine on
RewriteRule ^(/.*)$ https://%{HTTP_POST}$1
[redirect=301]
</VirtualHost>
```

Restart the httpd service, using systemctl start httpd.service and make sure it is enabled by using systemctl enable httpd.service.

After all, setting up a web server that uses TLS for enhanced security may seem difficult, but It is not. Once you have access to the public/private key pair, you only need to configure the SSLCertificateFile and the SSLCertificateKeyFile directives to tell Apache where these files can be found. You do not need to remember any syntax specifics either because everything you need is in the mod_ssl package that needs to be installed to use TLS.

Quiz

Chapter review questions

1. On default RHEL7 installation, Which command installs the software packages that are needed to configure an Apache web server?
   a. yum install httpd
   b. yum install web-server
   c. yum install apache
   d. yum install apache2
2. What is the name of the main Apache configuration file?
   a. /etc/httpd/conf/httpd. conf
   b. /etc/httpd/httpd. conf
   c. /etc/apache2/apache.conf
   d. /etc/httpd/default -server.conf
3. Which parameter in the Apache configuration file is used to specify where Apache will serve its documents from?
   a. ServerRoot
   b. ServerDocuments
   c. DocumentRoot
   d. DocumentIndex
4. Which parameter in the main Apache configuration file defines the location where the Apache process looks for its configuration files?
   a. ServerRoot
   b. ServerDocuments
   c. DocumentRoot
   d. DocumentIndex

5. Which directory contains the main Apache configuration file?
   a. /etc/httpd
   b. /etc/htttpd/conf
   c. /etc/httpd/conf.d
   d. /etc/httpd/conf. modules.d
6. Which directory contains the configuration files for the different Apache modules?
   a. /etc/httpd
   b. /etc/htttpd/conf
   c. /etc/httpd/conf.d
   d. /etc/httpd/conf.modules.d
7. Which directory is used to drop configuration files that are installed from RPMs?
   a. /etc/httpd
   b. /etc/htttpd/conf
   c. /etc/httpd/conf.d
   d. /etc/httpd/conf.modules.d
8. Which virtual host type allows you to run multiple virtual hosts on the same IP address?
   a. NameBased
   b. IPBased
   c. ConfigurationBased
   d. Default
9. Which line is used to start the definition of a virtual host that listens on port 80 of all IP addresses on the current server?
   a. <VirtualHost *: 80>
   b. <VirtualHost *>
   c. <NameHost *: 80>
   d. <NameHost *>

10. Which of the following statements about virtual hosts is not true?
    a. When virtual hosts are offered through an httpd process, the default configuration no longer works.
    b. The names of virtual hosts must be resolvable through /etc/hosts or DNS.
    c. To use virtual hosts, the mod_virt package must be installed.
    d. Virtual host configurations can be specified in httpd.conf.
11. What specifically is the AllowOverride directive used for?
    a. If set to yes, the contents of a directory can be changed.
    b. This setting is specifically for user home directories. If set to yes, these will be included in the Apache configuration.
    c. If set to yes, the htaccess file in Apache directories will be considered, which will have a performance price.
    d. Set to yes if you want to allow users to create additional configuration files in directories.
12. Which utility enables you to generate a TLS certificate and key for setting up a TLS security Apache web server?
    a. genkey
    b. openssl
    c. createkey
    d. Sslkey
13. What is the default directory where the TLS private key is stored?
    a. /etc/ssl/certs/servername.key
    b. /etc/pki/tls/private/servername.key
    c. /etc/tls/keys/servername.key
    d. /etc/ssl/certs/private/servername.key

14. When configuring an Apache server for use of TLS, some directives are generally changed. Which of the following is not typically among them?

a. SSLEngine

b. SSLCertificateFile

c. SSLCertificateKeyFile

d. ServerName

15. How do you enable the httpd service to be started automatically when Booting?

a. systemctl enable httpd

b. systemctl start httpd

c. systemctl httpd enable

d. Service enable httpd

16. Which command enables you to test a web server from a server that does not offer a graphical interface?

a. scp

b. ssh

c. elinks

d. firefox

17. What is the name of the default Apache configuration file?

a. http.conf

b. apache.conf

c. httpd.conf

d. conf.d

18. Which command enables you to see whether the Apache web server is currently running?

a. systemctl status apache

b. systemctl apache status

c. systemctl httpd status

d. systemctl status httpd

19. In which directory would you typically find the TLS certificate file and the TLS key?

a. /etc/certs/pki

b. /var/log/certs/

c. /etc/pki/tls/certs/

d. /etc/certs/

20. What is the default configuration file for configuring TLS secured web servers?

a. ssl.conf

b. httpd.conf

c. apache.conf

d. http.conf

Answers to chapter Review Questions:

1. a
2. b
3. c
4. a
5. b
6. d
7. a
8. a
9. a
10. c
11. c
12. a
13. b
14. d
15. a
16. a
17. c
18. d
19. c
20. a