

4. Administración de Apache

4.19. Servidor virtual HTTPS por defecto en *Linux*

Realiza la siguiente configuración en el servidor *Apache* instalado en **ServidorLinuxXX**.

- Habilita el servidor virtual por defecto.
- Deshabilita los servidores virtuales creados en las prácticas anteriores.
- Habilita el modulo *mod_ssl*.
- *Habilita el servidor virtual *ssl* por defecto.*

Prueba la configuración.

1. Inicia una sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.
2. Habilita el servidor virtual por defecto de *Apache*.

```
sudo a2ensite 000-default
```

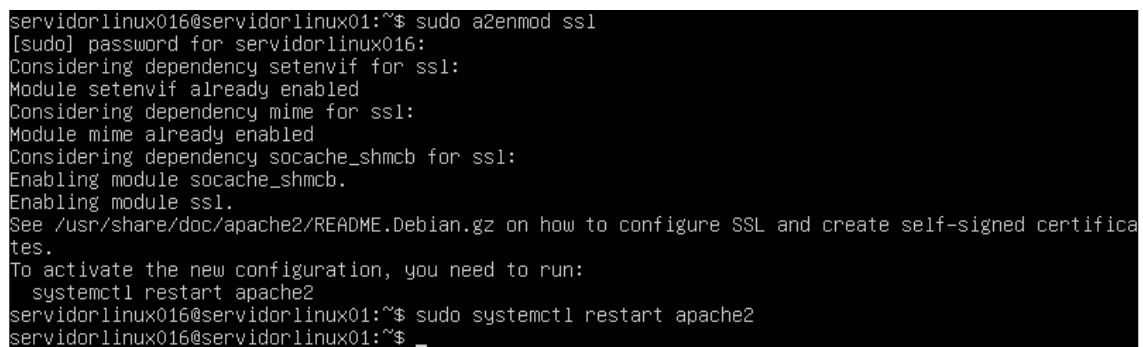
3. Verifica que dentro del directorio **/etc/apache2/sites-enabled** se ha creado el enlace **000-default**.
4. Deshabilita los servidores virtuales creados en prácticas anteriores.

```
sudo a2dissite software
```

```
sudo a2dissite hardware
```

5. Reinicia el servidor para que los cambios tengan efecto.
6. Habilita el módulo modssl que permite usar https, Figura 4.110.

```
sudo a2enmod ssl
```



```
servidorlinux016@servidorlinux01:~$ sudo a2enmod ssl
[sudo] password for servidorlinux016:
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certificates.
To activate the new configuration, you need to run:
  systemctl restart apache2
servidorlinux016@servidorlinux01:~$ sudo systemctl restart apache2
servidorlinux016@servidorlinux01:~$ _
```

Figura 4.110: Habilitar el modulo modssl

7. Reinicia el servidor para que los cambios tengan efecto
8. Consulta el fichero **/etc/apache2/port.conf** y observa que si habilita el modulo *ssl* el servidor escuchará en el puerto 443. Figura 4.111

```

servidorlinux016@servidorlinux01:~$ sudo cat /etc/apache2/ports.conf
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 80

<IfModule ssl_module>
    Listen 443
</IfModule>

<IfModule mod_gnutls.c>
    Listen 443
</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
servidorlinux016@servidorlinux01:~$

```

Figura 4.111: Escucha el puerto 443

9. Verifica que el servidor escucha en los puertos 80/TCP y 443/TCP

netstat -ltn

10. Accede al directorio **/etc/apache2/sites-available** y observa que existe un fichero denominado **default-ssl.conf** que contiene la configuración por defecto de un servidor HTTPS.
11. Habilita el servidor virtual ssl defecto (default-ssl) de *Apache*.

sudo a2ensite default-ssl.conf

12. Reinicia el servidor para que los cambios tengan efecto.
13. Consulta el fichero **/etc/apache2/sites-available/default-ssl.conf** y observa su configuración. Fíjate en las directivas que habilitan SSL y que definen la ruta del certificado digital que usará el servidor, Figura 4.112 y 4.113.

El servidor utiliza por defecto un certificado digital autofirmado que se ha creado al instalar Apache. Un certificado autofirmado no está firmado por una autoridad de certificación (tercera parte de confianza) y por tanto, no existen mecanismos automáticos que garanticen su autenticidad. Por eso los navegadores nos pedirán confirmación cuando el servidor se lo envíe.

```

<IfModule mod_ssl.c>
    <VirtualHost _default_:443>
        ServerAdmin webmaster@localhost

        DocumentRoot /var/www/html

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf

        # SSL Engine Switch:
        # Enable/Disable SSL for this virtual host.
        SSLEngine on

        # A self-signed (snakeoil) certificate can be created by installing
        # the ssl-cert package. See
        # /usr/share/doc/apache2/README.Debian.gz for more info.
        # If both key and certificate are stored in the same file, only the
        # SSLCertificateFile directive is needed.
        SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem
        SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
    
```

Figura 4.112: Fichero **/etc/apache2/sites-available/default-ssl.conf**

```

# Server Certificate Chain:
# Point SSLCertificateChainFile at a file containing the
# concatenation of PEM encoded CA certificates which form the
# certificate chain for the server certificate. Alternatively
# the referenced file can be the same as SSLCertificateFile
# when the CA certificates are directly appended to the server
# certificate for convenience.
#SSLCertificateChainFile /etc/apache2/ssl.crt/server-ca.crt

# Certificate Authority (CA):
# Set the CA certificate verification path where to find CA
# certificates for client authentication or alternatively one
# huge file containing all of them (file must be PEM encoded)
# Note: Inside SSLCACertificatePath you need hash symlinks
#       to point to the certificate files. Use the provided
#       Makefile to update the hash symlinks after changes.
#SSLCACertificatePath /etc/ssl/certs/
#SSLCACertificateFile /etc/apache2/ssl.crt/ca-bundle.crt

# Certificate Revocation Lists (CRL):
# Set the CA revocation path where to find CA CRLs for client
# authentication or alternatively one huge file containing all
# of them (file must be PEM encoded)
# Note: Inside SSLCARevocationPath you need hash symlinks
#       to point to the certificate files. Use the provided
#       Makefile to update the hash symlinks after changes.
#SSLCARevocationPath /etc/apache2/ssl.crl/
#SSLCARevocationFile /etc/apache2/ssl.crl/ca-bundle.crl

# Client Authentication (Type):
# Client certificate verification type and depth. Types are
# none, optional, require and optional_no_ca. Depth is a
# number which specifies how deeply to verify the certificate

```

Figura 4.113: Fichero `/etc/apache2/sites-available/default-ssl.conf`

```

# number which specifies how deeply to verify the certificate
# issuer chain before deciding the certificate is not valid.
#SSLVerifyClient require
#SSLVerifyDepth 10

# SSL Engine Options:
# Set various options for the SSL engine.
# o FakeBasicAuth:
#   Translate the client X.509 into a Basic Authorisation. This means that
#   the standard Auth/DBMAuth methods can be used for access control. The
#   user name is the 'one line' version of the client's X.509 certificate.
#   Note that no password is obtained from the user. Every entry in the user
#   file needs this password: 'xxj312MTZzKVA'.
# o ExportCertData:
#   This exports two additional environment variables: SSL_CLIENT_CERT and
#   SSL_SERVER_CERT. These contain the PEM-encoded certificates of the
#   server (always existing) and the client (only existing when client
#   authentication is used). This can be used to import the certificates
#   into CGI scripts.
# o StdEnvVars:
#   This exports the standard SSL/TLS related 'SSL_*' environment variables.
#   Per default this exportation is switched off for performance reasons,
#   because the extraction step is an expensive operation and is usually
#   useless for serving static content. So one usually enables the
#   exportation for CGI and SSI requests only.
# o OptRenegotiate:
#   This enables optimized SSL connection renegotiation handling when SSL
#   directives are used in per-directory context.
#SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
<FilesMatch "\.(cgi|shtml|phtml|php)$">
    SSLOptions +StdEnvVars
</FilesMatch>
<Directory /usr/lib/cgi-bin>

```

Figura 4.113-b: Fichero `/etc/apache2/sites-available/default-ssl.conf`

```

<Directory /usr/lib/cgi-bin>
    SSLOptions +StdEnvVars
</Directory>

# SSL Protocol Adjustments:
# The safe and default but still SSL/TLS standard compliant shutdown
# approach is that mod_ssl sends the close notify alert but doesn't wait for
# the close notify alert from client. When you need a different shutdown
# approach you can use one of the following variables:
# o ssl-unclean-shutdown:
#     This forces an unclean shutdown when the connection is closed, i.e. no
#     SSL close notify alert is send or allowed to received. This violates
#     the SSL/TLS standard but is needed for some brain-dead browsers. Use
#     this when you receive I/O errors because of the standard approach where
#     mod_ssl sends the close notify alert.
# o ssl-accurate-shutdown:
#     This forces an accurate shutdown when the connection is closed, i.e. a
#     SSL close notify alert is send and mod_ssl waits for the close notify
#     alert of the client. This is 100% SSL/TLS standard compliant, but in
#     practice often causes hanging connections with brain-dead browsers. Use
#     this only for browsers where you know that their SSL implementation
#     works correctly.
# Notice: Most problems of broken clients are also related to the HTTP
# keep-alive facility, so you usually additionally want to disable
# keep-alive for those clients, too. Use variable "nokeepalive" for this.
# Similarly, one has to force some clients to use HTTP/1.0 to workaround
# their broken HTTP/1.1 implementation. Use variables "downgrade-1.0" and
# "force-response-1.0" for this.
# BrowserMatch "MSIE [2-6]" \
#     nokeepalive ssl-unclean-shutdown \
#     downgrade-1.0 force-response-1.0

</VirtualHost>

```

Figura 4.113-c: Fichero `/etc/apache2/sites-available/default-ssl.conf`

14. Desde **DesarrolloW7XX** abre el navegador y establece una conexión a `http://172.16.10.XX6`, Figura 4.114



Figura 4.114: Conexión http

15. Desde **DesarrolloW7XX** abre el navegador y establece una conexión a `https://172.16.10.XX6`, Figuras 4.115, 4.116.

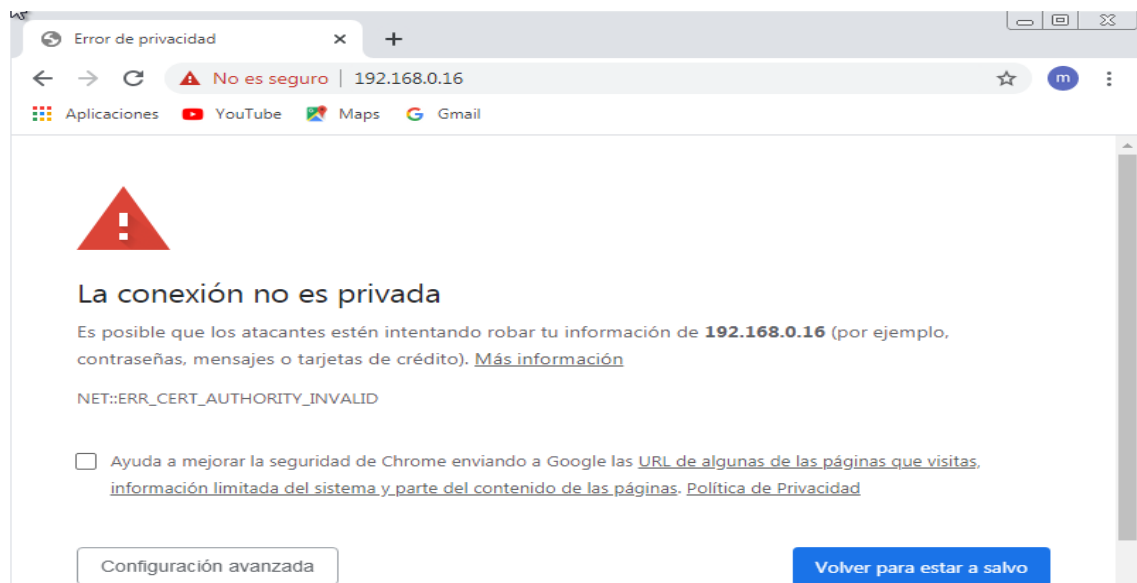


Figura 4.115: Conexión https



Figura 4.116: Conexión https

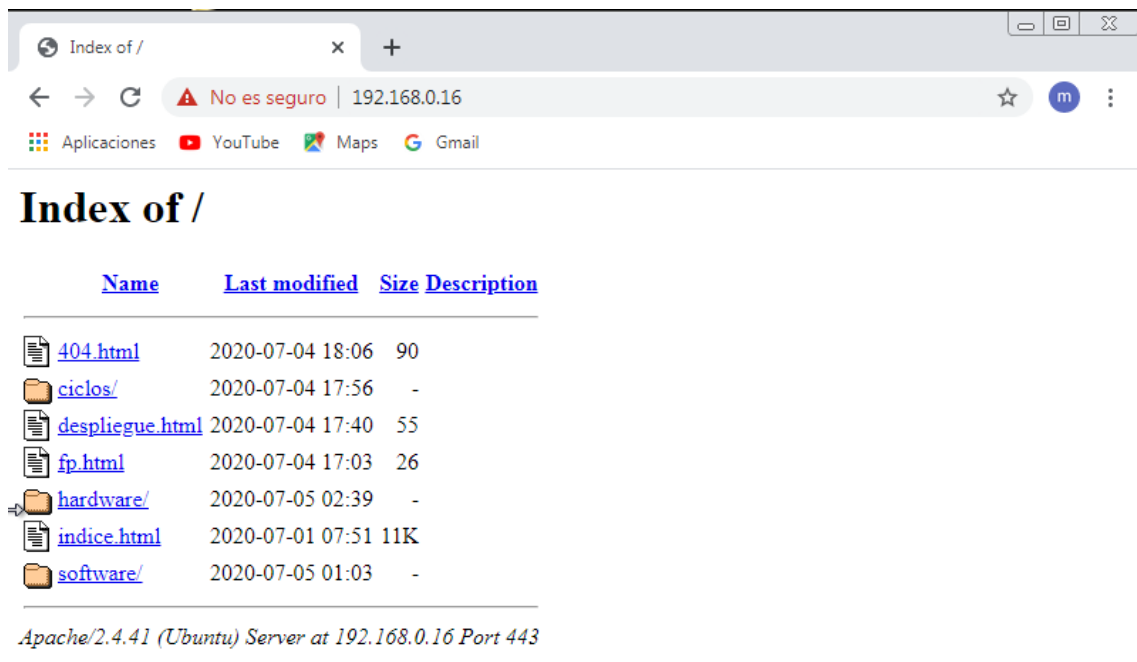


Figura 4.116-b: Conexión https