MEDAC
Instituto Oficial de Formación Profesional

DESPLIEGUE DE APLICACIONES WEB

TÉCNICO EN DESARROLLO DE APLICACIONES WEB

Autenticación y control de acceso

05

ÍNDICE

 ISBN: 978-84-18983-26-9

OBJETIVOS

Aprender a generar y utilizar certificados autofirmados.

Utilizar herramientas externas para obtener certificados válidos.

Implementar un servidor de certificación.

Aprender a securizar los accesos de un servidor.

Configurar sistemas de autenticación de usuarios.

/ 1. Introducción y contextualización práctica

En este tema, nos centraremos en la seguridad. Veremos cómo generar e instalar certificados firmados por una Autoridad de Certificación (presente como AC en castellano, o CA en inglés: Certification Authority) para cifrar las conexiones entre el cliente y el servidor.

También nos enfocaremos en securizar los accesos al propio servidor y a los recursos compartidos. Para ello, utilizaremos y aprenderemos sobre sistemas de control de acceso como pueden ser los cortafuegos.

Planteamiento del caso práctico inicial

A continuación, vamos a plantear un caso a través del cual podremos aproximarnos de forma práctica a la teoría de este tema.

Escucha el siguiente audio donde planteamos la contextualización práctica de este tema, encontrarás su resolución en el apartado Resumen y resolución del caso práctico

Fig. 1. Departamento de Atención al Cliente.

Audio intro. "Seguridad en la empresa
https://bit.ly/2ECqV30

/ 2. Certificados SSL

Un certificado SSL (Secure Sockets Layer), también llamado certificado digital, a secas (aunque esto pueda generar confusión con otros tipos de certificados digitales existentes), es el **"documento" que acredita la identidad de un servidor web.**

El uso de los certificados SSL permiten establecer una conexión segura con el servidor.

Las ventajas de usar un certificado digital son:

- Permiten verificar que el sitio web al que estamos accediendo es legítimo. Esto es posible realizando una validación entre el nombre DNS y la dirección IP del servidor al que apunta.
- Realizar un cifrado de los datos transmitidos utilizando la clave privada del certificado.

En casos prácticos de temas anteriores, hemos aprendido el proceso de instalación de certificados autofirmados para utilizar el protocolo HTTPS. Al cifrar la comunicación entre el cliente y el servidor, garantizamos la integridad de la conexión. Al no poder comprobar la identidad del servidor, recordamos que Google Chrome reconocía la conexión como insegura.

Este problema se soluciona utilizando un certificado firmado por una CA.

Las **CA** son organizaciones que se encargan de verificar la identidad de los solicitantes de un certificado, así como de la emisión de los mismos. También gestionan el estado de los certificados emitidos.

Entre las CA emisoras de certificados SSL más reputadas, podemos encontrar las siguientes:

- Comodo SSL
- GeoTrust SSL
- Symantec SSL

Aunque la primera razón para utilizar los certificados digitales es securizar las conexiones con nuestro servidor web, es importante remarcar que los buscadores web penalizan el uso de certificados digitales no firmados dando menor puntuación para SEO.

Fig. 2. Los certificados SSL garantizan la seguridad de la conexión.

Audio 1. "Certificados comodín"
https://bit.ly/3ga6tEl

2.1. Servidores de certificados

Para ciertos desarrollos, puede ser interesante disponer de una CA propia para generar certificados para nuestros servidores. A ojos de los navegadores, estos certificados no serán seguros, pues la identidad del servidor no se puede contrastar. Solo serán seguros si son emitidos por una CA "oficial". Sin embargo, si los instalamos en un cliente, este sí confiará en nuestro servidor web.

El primer paso para que nuestra CA con OpenSSL sea funcional es tener configurado un FQDN. Para ello, editamos el fichero /etc/hosts y modificamos la línea 127.0.0.1 añadiendo el nombre del servidor con el sufijo DNS de nuestro dominio.

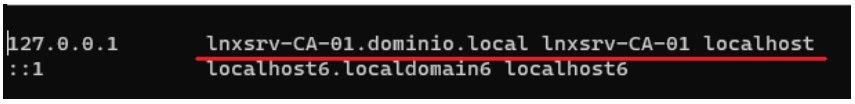

Fig. 3. Fichero hosts.

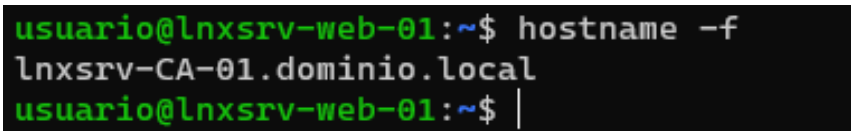

Fig. 4. Comprobación FQDN Linux.

OpenSSL establece la ruta para almacenar los certificados en "./**demoCA**". Para cambiar este parámetro y seleccionar una carpeta segura, modificaremos en el archivo **/usr/lib/ssl/openssl.cnf** la línea "*dir*" debajo del bloque *CA_default*.

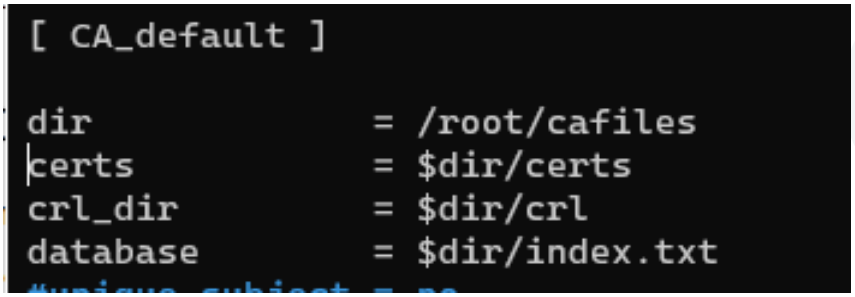

Fig. 5. Establecemos el directorio de los certificados en /root/cafiles

A continuación, crearemos la estructura de carpetas y los archivos index.txt y serial, que OpenSSL utilizará para realizar un seguimiento de los certificados creados. En este punto, ya podríamos crear la clave privada que utilizará nuestra CA.

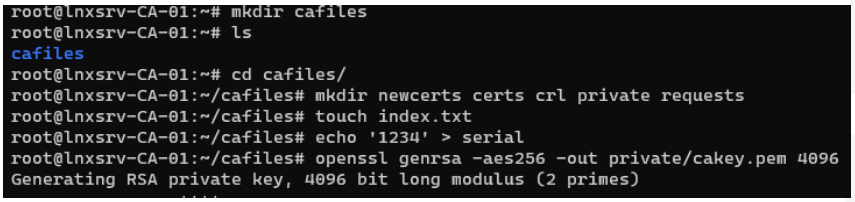

Fig. 6. Creación del certificado de la CA.

Por último, utilizaremos la clave privada para generar el certificado de la CA que nos servirá para firmar certificados de otros servidores.

```
root@lnxsrv-CA-01:~/cafiles# openssl req -new -x509 -key /root/cafiles/
private/cakey.pem -out cacert.pem -days 3650 -set
_serial 0
Enter pass phrase for /root/cafiles/private/cakey.pem:
```

Código 1. Creación del certificado de la CA.

En el siguiente enlace, podrás encontrar los pasos completos para instalar una CA y firmar certificados para otros servidores:

https://networklessons.com/uncategorized/openssl-certification-authority-ca-ubuntu-server

/ 3. Let's Encrypt

Let's Encrypt es una CA que permite generar certificados SSL de forma gratuita. En su web oficial se definen como "una Autoridad de Certificación gratuita, automatizada, y abierta".

Los certificados generados con Let's Encrypt tendrán una validez de tres meses. Pasado este tiempo, tendremos que realizar la renovación.

El proceso de instalación y configuración se realiza de manera automática utilizando las diferentes opciones que Let's Encrypt ofrece, aunque la forma que la propia CA recomienda es Certbot, que veremos a continuación.

En el siguiente enlace encontrarás el listado completo de formas de validación:

https://letsencrypt.org/es/docs/client-options/

Fig. 7. Logo Let's Encrypt.

3.1. Cerbot

Certbot es una herramienta Open Source utilizada para automatizar la instalación de certificados SSL de Let's Encrypt en servidores web.

En la página oficial de Certbot, encontraremos las instrucciones para realizar la instalación dependiendo del servidor web utilizado y del sistema operativo.

https://certbot.eff.org/instructions

Una vez instalado, Certbot es capaz de añadir las líneas necesarias al Virtual Host correspondiente, además de encargarse de renovar el certificado automáticamente cuando sea necesario. Para ello, Certbot añade una tarea programada en el fichero crontab.

Fig. 8. Logo Certbot.

/ 4. Caso práctico 1: "Certbot"

Planteamiento: Pedro trabaja en una empresa que tiene algunos servidores para uso interno y expuestos a internet, pero no quieren afrontar el gasto extra en certificados SSL firmados.

Están buscando la solución más económica posible, ya que en muchos casos estos servidores son temporales y a los pocos meses se desactivan.

Nudo: ¿Qué solución plantearías? ¿Necesitará crear certificados autofirmados?

Desenlace: En este caso, y teniendo en cuenta que *Certbot* es gratuito, Pedro podría utilizarlo para instalar certificados SSL firmados y así eliminar la alerta de seguridad en los navegadores sin tener que realizar una inversión en la compra de un certificado.

Para instalar *Certbot*, Pedro puede utilizar el asistente de la web oficial, donde encontrará las instrucciones para aplicarlo en su servidor Apache con Ubuntu 18.

Paso 1: añadir el repositorio de Certbot.

```
sudo apt-get update
sudo apt-get install software-properties-common
sudo add-apt-repository universe
sudo add-apt-repository ppa:certbot/certbot
sudo apt-get update
```

Código 2.1. Add CertBot PPA. Extraido de https://certbot.eff.org/

Paso2: instalar Certbot.

```
sudo apt-get install certbot python3-certbot-apache
```

Código 2.2. Instalar Certbot. Extraido de https://certbot.eff.org/

Paso3: ejecutar Certbot.

```
sudo certbot --apache
```

Código 2.3. Crear certificado. Extraido de https://certbot.eff.org/

/ 5. Acceso a carpetas seguras

Del mismo modo que aseguramos la conexión entre el cliente y el servidor instalando certificados digitales y haciendo uso del protocolo HTTPS, también es importante realizar una buena configuración de nuestro servidor para limitar los accesos y evitar que una carpeta con información sensible esté expuesta a todo internet.

En este apartado hablaremos de dos formas de limitar el acceso a carpetas: mediante el uso del fichero .htaccess o con Virtual Hosts.

En cualquier caso, es necesario habilitar la directiva **AllowOverride All** en el fichero *Apache.conf*.

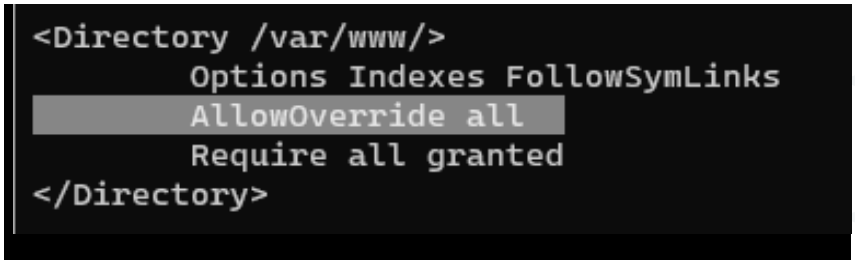

Fig. 9. Habilitando el uso de htaccess en /etc/apache2/apache.conf

5.1. htaccess

Según Apache:

> Los ficheros `.htaccess` (o "ficheros de configuración distribuida") facilitan una forma de realizar cambios en la configuración en contexto directorio. Un fichero, que contiene una o más directivas, se coloca en un documento específico de un directorio, y estas directivas aplican a ese directorio y todos sus subdirectorios.

Fig. 10. Definición de los ficheros .htaccess extracto de la web de Apache

https://httpd.apache.org/docs/2.4/es/howto/htaccess.html.

Para limitar el acceso total o de una determinada dirección IP, crearemos un archivo *.htaccess* en la carpeta que queremos limitar con el siguiente contenido:

Deny from [All | IP]

Al recargar Apache, la web de la carpeta solicitada nos devolverá un error 403.

5.2. Virtual Host

Debido a que puede afectar al rendimiento del servidor, Apache recomienda no utilizar los ficheros .htaccess. En su lugar, es más efectivo realizar las redirecciones o bloqueos utilizando Virtual Host.

La seguridad en los Virtual Hosts se define utilizando la estructura <Directory>.

```
<VirtualHost *:80>
        ServerName www.apache1.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/web1
<Directory "/var/www/web1">
Deny from all
</Directory>
</VirtualHost>
```

Fig. 11. Ejemplo de virtualHost denegando acceso a directorio web1.

Vídeo 1. "Securizar carpetas utilizando Virtual Hosts"

https://bit.ly/30bSeJU

/ 6. Autenticación de usuarios

En Apache, existen mecanismos para proteger ficheros y directorios utilizando autenticación con contraseña. Esta configuración se realizará desde los Virtual Hosts. De todos estos mecanismos, nos centraremos en dos tipos: *Basic y Digest*.

6.1. Basic

Este sistema de autenticación es el más simple y requiere el módulo *mod_authbasic* para su funcionamiento. Las contraseñas se transmitirán en texto plano.

Para activarlo, tendremos que añadir en el fichero de configuración del sitio web las siguientes líneas:

```
<Directory "/var/www/carpeta-segura">
    Order deny,allow
    AuthUserFile "/etc/apache2/basicAuth.txt"
    AuthName "Contraseña"
    AuthType Basic
    Require valid-user
</Directory>
```

Fig. 12. Ejemplo de virtualHost con autenticación Basic.

A continuación, añadiremos el usuario especificado en el Virtual Host, guardando el usuario y la contraseña en fichero basicAuth.txt.

```
usuario@lnxsrv-web-01:~$ sudo htpasswd -c /etc/apache2/basicAuth.txt usuari
New password:
```

Fig. 13. Creación del fichero de autenticación del método Basic.

Tras reiniciar Apache, el servidor solicitará usuario y contraseña para acceder a esta carpeta.

En la web de Apache, encontraremos más información al respecto:

https://httpd.apache.org/docs/2.4/es/mod/mod_auth_basic.html

6.2. Digest

Si utilizamos el método de autenticación Digest en sitios seguros HTTPS, las contraseñas irán cifradas. Este tipo de autenticación es aún más segura que Basic.

Para que la autenticación Digest funcione correctamente, es necesario activar el módulo auth_Digest.

```
usuario@lnxsrv-web-01:~$ sudo a2enmod auth_digest
Considering dependency authn_core for auth_digest:
Module authn_core already enabled
Enabling module auth_digest.
To activate the new configuration, you need to run:
  systemctl restart apache2
usuario@lnxsrv-web-01:~$ sudo apache2ctl restart
```

Fig. 14. Activación del módulo auth_digest.

Posteriormente, crearemos el archivo con las contraseñas y daremos permisos y propiedad del archivo al usuario *www-data:*

```
usuario@lnxsrv-web-01:~$ sudo htdigest -c /etc/apache2/authDigest digest usuarioDigest
Adding password for usuarioDigest in realm digest.
New password:
Re-type new password:
usuario@lnxsrv-web-01:~$ sudo chmod 640 /etc/apache2/authDigest
usuario@lnxsrv-web-01:~$ sudo chown root.www-data /etc/apache2/authDigest
```

Fig. 15. Creación del fichero de contraseñas para el método Digest.

Por último, añadiremos la configuración al Virtual Host y comprobaremos el funcionamiento.

```
Alias /restringido /var/www/carpeta_digest/
<Directory "/var/www/carpeta_digest">
    AuthType Digest
    AuthName "Restringido"
    AuthDigestProvider file
    AuthUserFile /etc/apache2/Authdigest
    Require user usuarioDigest usuario2
</Directory>
```

Fig. 16. Ejemplo de virtualHost con autenticación Digest.

/ 7. Mecanismos de control de acceso: UFW

Otra capa de seguridad que añadiremos al servidor serán **controles de acceso** para limitar ciertos servicios por IP y/o proteger los puertos de gestión de ataques de fuerza bruta.

UFW es un *software* que **realiza funciones de cortafuegos en sistemas Linux.** Es el acrónimo de *Uncomplicated Firewall*. Está diseñado como una interfaz sencilla para gestionar *iptables*.

UFW nos permite denegar acceso a determinados puertos o bloquear tráfico que proceda de una dirección IP concreta. Del mismo modo, se puede utilizar para establecer conexiones seguras, permitiendo, exclusivamente, las direcciones de Internet (IP) que nos interesen.

En muchas distribuciones de Linux, UFW viene instalado por defecto. En caso contrario, podemos instalarlo ejecutando el comando **apt-get install ufw.**

```
usuario@lnxsrv-web-01:~$ sudo ufw status
Status: inactive
usuario@lnxsrv-web-01:~$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
usuario@lnxsrv-web-01:~$
```

Fig. 17. Comprobación de estado del firewall de Linux y activación.

Comando	Descripción
ufw status numbered	Muestra el estado del fw y un listado numerado de las reglas
ufw enable	Activa el firewall
ufw disable	Desactiva el firewall
ufw reload	Recarga el firewall para aplicar cambios
ufw allow	Añade una regla que permita tráfico
ufw deny	Añade una regla que deniegue tráfico

Tabla 1. Comprobación de estado del firewall de Linux y activación.

Algunos ejemplos de reglas que podríamos implementar en UFW serían las siguientes:

```
sudo ufw deny out to 1.1.1.1 comment 'denegar tráfico de salida a 1.1.1.1'

usuario@lnxsrv-web-01:~$ ping 1.1.1.1

PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.

ping: sendmsg: Operation not permitted

ping: sendmsg: Operation not permitted

sudo ufw status numbered

Status: active

     To                         Action      From

     --                         ------      ----

[ 1] 1.1.1.1                    DENY OUT    Anywhere                   (out) # denegar tráfico de salida a 1.1.1.1

sudo ufw delete 1

sudo ufw allow from 12.23.34.5 to any port 22 comment 'Permitir tráfico SSH desde una IP'
```

Código 3. Ejemplos de reglas.

/ 8. Caso práctico 2: "Conexión segura con servidores web"

Planteamiento: Pedro está configurando una aplicación web de facturación para un cliente y uno de los requisitos es maximizar la seguridad y que solo se pueda acceder desde la oficina o conectados a la VPN a modo de intranet.

El servidor se ha instalado sobre un VPS con Debian 9 y utiliza Nginx.

Nudo: ¿Cómo lo hará?

Desenlace: La solución más sencilla y económica es utilizar el cortafuegos de Linux para permitir, únicamente, la dirección IP de la empresa.

La IP pública de la VPN será la misma que en la oficina central, ya que el firewall principal está allí.

Además de limitar el acceso a la IP de la oficina del cliente, también deberá permitirse acceso para tener gestión del servidor. Un error en las reglas creadas podría hacer que el servidor quedase incomunicado.

Mostramos un ejemplo de una posible configuración:

```
# sudo ufw allow from 46.56.12.5 to any port 80 comment 'Permitir tráfico HTTP SEDE'
Rule added
# sudo ufw allow from 46.56.12.5 to any port 443 comment 'Permitir tráfico HTTPS SEDE'
Rule added
# sudo ufw allow from 15.25.58.88 to any port 80 comment 'Permitir tráfico HTTP GESTION'
Rule added
# sudo ufw allow from 15.25.58.88 to any port 443 comment 'Permitir tráfico HTTPs GESTION'
Rule added
# sudo ufw allow from 15.25.58.88 to any port 22 comment 'Permitir tráfico SSH GESTION'
Rule added
# ufw status numbered
Status: active
     To                         Action      From
     --                         ------      ----
[ 1] 443                        ALLOW IN    46.56.12.5                 # Permitir tráfico HTTPS SEDE
[ 2] 80                         ALLOW IN    46.56.12.5                 # Permitir tráfico HTTP SEDE
[ 3] 80                         ALLOW IN    15.25.58.88                # Permitir tráfico HTTP GESTION
[ 4] 443                        ALLOW IN    15.25.58.88                # Permitir tráfico HTTPs GESTION
[ 5] 22                         ALLOW IN    15.25.58.88                # Permitir tráfico SSH GESTION
```

Código 4. Configuración UFW.

/ 9. Mecanismos de control de acceso: Fail2Ban

Fail2Ban es un **sistema de protección contra ataques de fuerza bruta** escrito en Python. Si detecta un número de intentos de inicio de sesión, bloquea la dirección IP durante un determinado tiempo que podemos definir en su configuración.

Este software analiza los logs de Apache y banea direcciones sospechosas. Fail2Ban nos ayuda a proteger los puertos de gestión, como podría ser el 22 (SSH), pero también los puertos 80 y 443.

Fig. 18. Logo Fail2Ban.

Podemos encontrar más información en la web oficial de Fail2Ban: https://www.fail2ban.org/wiki/index.php/Main_Page

La instalación de Fail2Ban en Ubuntu se realizará ejecutando el comando apt-get install fail2ban.

La configuración general de Fail2Ban se almacena en **/etc/fail2ban/fail2ban.conf**, pero, además, existen más archivos como *jail.conf* que guardan información de las "jaulas" o monitores que tenemos activos.

Lo recomendable es no editar el archivo jail.conf. En su lugar, debemos crear un nuevo fichero llamado *jail.local,* donde introduciremos la configuración de las nuevas jaulas

```
usuario@lnxsrv-web-01:~$ sudo cat /etc/fail2ban/jail.local
[sudo] password for usuario:
[sshd]
enabled = true
port = 22
filter = sshd
logpath = /var/log/auth.log
maxretry = 3
```

Fig. 19. Ejemplo de jaula para limitar el intento de inicio de sesión SSH a 3.

```
usuario@lnxsrv-web-01:~$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 1
|  |- Total failed:     4
|  `- File list:        /var/log/auth.log
`- Actions
   |- Currently banned: 0
   |- Total banned:     1
   `- Banned IP list:
usuario@lnxsrv-web-01:~$ sudo fail2ban-client set sshd unbanip DIRECCION_IP
 NOK: ('DIRECCION_IP is not banned',)
DIRECCION_IP is not banned
```

Fig. 20. Comprobar estado de filtro ssh y unban de IP.

Vídeo 2. "Fail2Ban"
https://bit.ly/3327g6N

/ 10. Resumen y resolución del caso práctico de la unidad

En esta unidad, hemos expuesto los mecanismos que podemos implementar para mejorar la seguridad en nuestro servidor, evitando así incidentes. Algunas de las opciones vistas han sido el uso del fichero .htaccess o los Virtual Hosts, que permiten impedir accesos a determinadas carpetas, o el establecimiento de contraseñas en directorios sensibles para limitar los usuarios que pueden visualizar ese contenido.

Fig. 21. Seguridad en directorios.

Además, hemos visto qué son las CA y cómo configurarlas para poder firmar nuestros propios certificados y, posteriormente, instalarlos en los clientes.

Finalmente, hemos hablado de **mecanismos de control de acceso** para proteger nuestro servidor de ataques o permitir, únicamente, conexiones de una determinada dirección IP.

Resolución del caso práctico inicial

En el caso práctico inicial, se plantea un problema de seguridad que permite acceder sin ningún tipo de restricciones a una carpeta que contiene información sensible.

La solución más acertada sería revisar la configuración de Apache y comprobar que la directiva AllowOverride está habilitada en ALL, y verificar que en el Virtual Host correspondiente se limita el acceso utilizando reglas restrictivas a cualquier IP, que no sea localhost.

```
<Directory /var/www/>
        Options Indexes FollowSymLinks
        AllowOverride all
        Require all granted
</Directory>
```

Fig. 22. Directiva AllowOverride en /etc/apache2/apache.conf.

/ 11. Webgrafía

https://www.osmosislatina.com/apache/vhosting.htm
https://www.ispconfig.org/
https://es.wikipedia.org/wiki/CPanel
https://www.digitalocean.com/community/tutorials/how-to-create-temporary-and-permanent-redirects-with-nginx
https://www.digitalocean.com/community/tutorials/how-to-set-up-nginx-server-blocks-virtual-hosts-on-ubuntu-16-04
https://httpd.apache.org/docs/2.4/es/howto/htaccess.html
https://httpd.apache.org/docs/trunk/es/howto/auth.html
https://www.zeppelinux.es/autenticacion-digest-en-apache/
https://networklessons.com/uncategorized/openssl-certification-authority-ca-ubuntu-server