31/10/2025

SECURIZACIÓN COMPLETA DE UN SERVIDOR WEB

Objetivos:

Configurar un servidor web Apache con HTTPS, autenticación de usuarios y control de acceso mediante firewall y Fail2Ban.

Adjunta captura de pantalla de cada punto.

Antes de empezar:

Requisitos previos:

- 2 máquinas virtuales Ubuntu:
    - Servidor web: Apache instalado.
    - Cliente: Navegador para verificar el acceso.
- Conexión en red (NAT o puente).

PARTE 1: Crear una CA propia con OpenSSL

Útil para prácticas de laboratorio y para entender cómo se firman certificados. Los navegadores **no** confiarán en esta CA por defecto salvo que instales el certificado raíz en el cliente.

Paso 1: En el servidor (o en una VM separada que actúe de CA):

```
sudo -i
mkdir -p /root/cafiles/{certs,crl,newcerts,private}
chmod 700 /root/cafiles/private
touch /root/cafiles/index.txt
echo 1000 > /root/cafiles/serial
```

Paso 2: Crea la clave privada de la CA (protegida con passphrase):

```
openssl genrsa -aes256 -out /root/cafiles/private/cakey.pem 4096
chmod 400 /root/cafiles/private/cakey.pem
```

Paso 3: Crea el certificado raíz (autofirmado) válido, por ejemplo, 10 años:

```
openssl req -x509 -new -nodes -key
/root/cafiles/private/cakey.pem \
  -sha256 -days 3650 -out /root/cafiles/cacert.pem \
  -subj "/C=ES/ST=Madrid/L=Leganes/O=MiCA/OU=IT/CN=mi-ca.local"
```

Paso 4: Firmar un certificado para un servidor (CSR → firmado por la CA)

En el servidor web:

```
# 1) crear clave del servidor
sudo openssl genrsa -out /etc/ssl/private/web01.key 2048

# 2) crear CSR (customer signing request)
```

```
sudo openssl req -new -key /etc/ssl/private/web01.key -out
/tmp/web01.csr \
  -subj
"/C=ES/ST=Madrid/L=Leganes/O=Empresa/OU=Web/CN=www.ejemplo.local
"

# copiar /tmp/web01.csr al host CA y firmar
# en la CA:
sudo openssl ca -in /root/cafiles/tmp/web01.csr -out
/root/cafiles/certs/web01.crt \
  -config /etc/ssl/openssl.cnf -batch
```

PARTE 2: Let's Encrypt + Certbot (recomendado para producción y práctica real)

Paso 1: Instalación y uso (Apache)

```
sudo apt update
sudo apt install software-properties-common -y
sudo add-apt-repository universe -y
sudo add-apt-repository ppa:certbot/certbot -y
sudo apt update
sudo apt install certbot python3-certbot-apache -y
```

Paso 2: Ejecuta Certbot (modo interactivo, añadirá config al virtualhost)

```
sudo certbot --apache
```

- Responderá con tu dominio (ej. `www.ejemplo.com`) y si deseas redirigir HTTP → HTTPS.
- Comprueba los certificados:

```
sudo certbot certificates
```

Comprueba renovación automática (systemd timer o crontab). Certbot normalmente añade renovación automática; verifica con sudo systemctl list-timers | grep certbot o sudo systemctl status certbot.timer.

PARTE 3: Pruebas

Desde cliente:

```
curl -I https://www.ejemplo.com
# o con detalle:
curl -vk https://www.ejemplo.com
```

Para inspeccionar certificado:

```
openssl s_client -connect www.ejemplo.com:443 -servername
www.ejemplo.com </dev/null 2>/dev/null | openssl x509 -noout -
text
```

PARTE 4: Acceso a carpetas seguras: .htaccess vs VirtualHost

Habilitar AllowOverride para permitir .htaccess. Edita /etc/apache2/apache2.conf y asegúrate de:

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

Luego:

```
sudo apache2ctl configtest
sudo systemctl reload apache2
```

* Cuidado: con `AllowOverride All` damos permiso a `.htaccess`

Opción 1 — bloqueo con `.htaccess`

Crea `/var/www/html/privada/.htaccess`:

```
# negar todo
Require all denied
# o para negar todo excepto una IP:
# Require not ip 192.168.1.0/24
```

Prueba → debe devolver 403. El PDF muestra el ejemplo `Deny from ...` (para versiones antiguas); en Apache 2.4 usar `Require` es mejor.

Opción 2 — bloqueo en VirtualHost (recomendado)

Ejemplo de VirtualHost (archivo `/etc/apache2/sites-available/ejemplo.conf`):

```
<VirtualHost *:80>
    ServerName www.ejemplo.local
    DocumentRoot /var/www/html/ejemplo

    <Directory /var/www/html/ejemplo/privada>
        Require all denied
        # permitir solo desde la intranet:
        # Require ip 192.168.1.0/24
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/ejemplo_error.log
    CustomLog ${APACHE_LOG_DIR}/ejemplo_access.log combined
</VirtualHost>
```

Habilitar y recargar:

```
sudo a2ensite ejemplo.conf
sudo apache2ctl configtest
sudo systemctl reload apache2
```

* usar <Directory> en VirtualHost por rendimiento

PARTE 5: Autenticación con contraseña en Apache (Basic y Digest)

A) Basic (mod_auth_basic)

1. Habilita modulo (normalmente instalado):

```
sudo a2enmod auth_basic
```

2. Crea el archivo de usuarios:

```
sudo htpasswd -c /etc/apache2/basicAuth.txt alumno1
# (te pedirá contraseña)
sudo chown root:www-data /etc/apache2/basicAuth.txt
sudo chmod 640 /etc/apache2/basicAuth.txt
```

3. Añade a VirtualHost (o <Directory>):

```
<Directory "/var/www/html/privada">
    AuthType Basic
    AuthName "Área restringida"
    AuthUserFile /etc/apache2/basicAuth.txt
    Require valid-user
</Directory>
```

4. Reinicia Apache y prueba desde navegador → pedirá credenciales.

Recuerda: Basic transmite credenciales en texto plano a no ser que uses HTTPS; por eso siempre usar sobre HTTPS o preferir Digest.

B) Digest (mod_auth_digest) — más seguro (se cifra)

1. Habilita módulo:

```
sudo a2enmod auth_digest
```

2. Crea fichero htdigest:

```
sudo htdigest -c /etc/apache2/digestAuth.txt "MiArea" alumno2
# -c solo la primera vez; para añadir más usuarios quitar -c
```

3. Config en VirtualHost:

```
<Directory "/var/www/html/privada">
    AuthType Digest
    AuthName "MiArea"
    AuthDigestProvider file
    AuthUserFile /etc/apache2/digestAuth.txt
    Require valid-user
```

```
</Directory>
```

4. Reinicia Apache y prueba (recuerda: Digest solo tiene sentido sobre HTTPS para mayor seguridad). El PDF señala que Digest requiere `auth_digest`.

Parte 6: UFW (Uncomplicated Firewall) — reglas y buenas prácticas

Instalación y reglas básicas:

```
sudo apt update
sudo apt install ufw -y

# permitir SSH desde cualquier IP (temporal, si vas a activar firewall):
sudo ufw allow 22/tcp

# permitir HTTP/HTTPS:
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp

# activar (cuidado: asegúrate SSH permitido antes)
sudo ufw enable

# ver estado:
sudo ufw status numbered
```

Permitir solo una IP para gestión:

```
sudo ufw allow from 15.25.58.88 to any port 22 comment 'Permitir SSH GESTION'
sudo ufw allow from 46.56.12.5 to any port 443 comment 'Permitir HTTPS SEDE'
```

Bloquear salida:

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

Importante: **antes de `ufw enable` asegúrate de que tienes una regla que permita SSH** desde tu IP actual para no perder acceso.

Parte 7: Fail2Ban para protección frente a fuerza bruta

Fail2Ban analiza logs y aplica bloqueos temporales. Se recomienda crear `jail.local` en lugar de editar `jail.conf`.

Instalación

```
sudo apt update
sudo apt install fail2ban -y
```

Configuración básica (archivo /etc/fail2ban/jail.local)

Crea `/etc/fail2ban/jail.local` con contenido mínimo:

```
[DEFAULT]
bantime  = 600
findtime = 600
```

```
maxretry = 3

[sshd]
enabled = true
port    = ssh
logpath = %(sshd_log)s
backend = systemd

[apache-auth]
enabled  = true
port     = http,https
filter   = apache-auth
logpath  = /var/log/apache2/*error.log
maxretry = 3
```

Reinicia:

```
sudo systemctl restart fail2ban
sudo fail2ban-client status
sudo fail2ban-client status sshd
```

Probar bloqueo

Desde otra máquina intenta varios intentos SSH fallidos; luego:

```
sudo fail2ban-client status sshd
# para desbloquear:
sudo fail2ban-client unban IP
```

Fail2Ban protege SSH, HTTP y HTTPS analizando logs y se sugiere no editar `jail.conf` sino usar `jail.local`.

PARTE 8: Tests y comprobaciones finales (qué entregar)

Añade en cada entrega:

1. **Capturas**:
   - `sudo certbot certificates` y `curl -vk https://...` mostrando certificado válido.
   - `apache2ctl configtest` y `systemctl status apache2`.
   - `ufw status numbered`.
   - `sudo fail2ban-client status` y `sudo fail2ban-client status sshd`.
2. **Captura de archivo de VirtualHost** usado (completo).
3. **Contenido** de `/etc/apache2/basicAuth.txt` (no la contraseña en claro: indicar usuario añadido y comprobación de acceso).
4. **Pruebas de vulnerabilidad sencillo**: mostrar que sin regla UFW todo acceso es posible; con regla UFW desde IP no permitida sale rechazado (captura de `ping`/`curl` fallando). El PDF emplea el caso de "Pedro" como ejemplo de restringir por IP.

Plantillas y archivos listos para entregar

A) VirtualHost ejemplo (HTTP -> redirigir a HTTPS si usas Certbot)

```
<VirtualHost *:80>
    ServerName www.ejemplo.local
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html/ejemplo

    ErrorLog ${APACHE_LOG_DIR}/ejemplo_error.log
    CustomLog ${APACHE_LOG_DIR}/ejemplo_access.log combined
</VirtualHost>
```

B) VirtualHost con `<Directory>` y autenticación Basic

```
<VirtualHost *:443>
    ServerName www.ejemplo.local
    DocumentRoot /var/www/html/ejemplo

    SSLEngine on
    SSLCertificateFile
/etc/letsencrypt/live/www.ejemplo.local/fullchain.pem
    SSLCertificateKeyFile
/etc/letsencrypt/live/www.ejemplo.local/privkey.pem

    <Directory /var/www/html/ejemplo/privada>
        AuthType Basic
        AuthName "Área restringida"
        AuthUserFile /etc/apache2/basicAuth.txt
        Require valid-user
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/ejemplo_ssl_error.log
    CustomLog ${APACHE_LOG_DIR}/ejemplo_ssl_access.log combined
</VirtualHost>
```

C) `jail.local` mínimo (Fail2Ban)

```
[DEFAULT]
bantime  = 600
findtime = 600
maxretry = 3

[sshd]
enabled = true
```

10) Errores comunes y consejos de evaluación

- **No habilitar SSH antes de `ufw enable`** → pierdes acceso.
- **Usar Basic sin HTTPS** → credenciales viajando en texto claro.
- **Editar jail.conf en vez de crear jail.local** → mala práctica; preferir `jail.local`.